

Smart Environment Based on Real-time Human Position Tracking for Remote Presence and Collaboration

Sachin Vasant Bharambe

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Benjamin R. Knapp
A. Lynn Abbott
Nicolas Polys

May 4, 2017
Blacksburg, Virginia

Keywords: Human Position Detection, Position Tracking, Network Communication,
Virtual Reality, Remote Presence and Collaboration

Copyright 2017, Sachin Bharambe

Smart Environment Based on Real-time Human Position Tracking for Remote Presence and Collaboration

Sachin Bharambe

ABSTRACT

Real-time, virtual and mixed reality systems have diverse uses for real-world data visualization, representation, and remote collaboration in distant learning settings, especially in universities. Design of such systems involves challenges in mapping the real world data and physical world structure accurately to digital form of physical space, also called as virtual models. Researchers have created similar systems using multiple cameras, stereo cameras, accelerometers, and motion detectors [1] [2] [3]. This report presents a platform to detect and track real-time locations of people present in buildings and map their location information into virtual models as avatars using omni-directional cameras installed in the physical space. These models were created as part of the Mirror Worlds project [4]. The project infrastructure is funded by National Science Foundation. This infrastructure enables users to connect virtual and physical aspects of the environment through a coordinate-based data networking system to enable interaction with the rest of the system including environment objects and other users [4]. This is an interdisciplinary project where students from various departments have worked on the development of virtual model of the Moss Art Center and Torgersen Hall in Unity / X3D. Some students from the Department of Computer Science have developed a coordinate-based data networking system. The prototype of a detection and tracking algorithm to extract the location information was developed using background subtraction in MATLAB.

The proposed approach was developed using the combination of background subtraction [5] and neural networks [6] along with heuristics based on spatial information about the physical space. The system was scaled to work across multiple buildings, extract the location information of people present in the physical space, and map location information into shared virtual space as an avatar. The concept of remote presence was extended to create a collaborative object manipulation application using Leap Motion controller. Effects of fidelity were evaluated to perform the collaborative object manipulation task in shared virtual

space based on user study conducted for this application.

Since no annotated people video dataset is publicly available with overhead view from omnidirectional cameras, three videos were annotated manually to test the performance of the approach. The current approach almost works at near real-time rates. All three video sequences were evaluated to compute frame based detection accuracy [7]. Precision and recall obtained for the first video sequence of people detection is 93.85% and 95.06% respectively.

Smart Environment Based on Real-time Human Position Tracking for Remote Presence and Collaboration

Sachin Bharambe

GENERAL AUDIENCE ABSTRACT

Globalization and the advancements in the technological world pose the need for people working across different geographical regions to communicate not just over video conferencing but having remote presence which provides a stimuli to users' senses to give the feeling of being at other location. The common approach to create remote presence application is by creating a real-time, virtual and mixed reality system. These mix reality systems have diverse uses for real-world data visualization, representation, and remote collaboration in distant learning settings, especially in universities. Design of such systems involves challenges in mapping the real world data and physical world structure accurately to digital form of physical space, also called as virtual models. Researchers have created similar systems using multiple cameras, depth sensors, accelerometers, and motion detectors.

This work introduces a platform to detect and track real-time locations of people present in buildings and map their location information into virtual models as avatars using cameras installed in the physical space. The proposed solution uses combination of background subtraction and neural networks with some heuristics based on spatial information about the physical space. The system is scaled to work across multiple buildings. The concept of remote presence was extended to create a collaborative object manipulation application.

Three videos were annotated manually to test the performance of the proposed approach.

Acknowledgement

Without the guidance and support of my committee members, team members, friends and family, my research would not have been possible. To begin, I would like to express my sincere gratitude and appreciation to Dr. Ben Knapp and Dr. Nicholas Polys for their invaluable support and guidance throughout my years at Virginia Tech. It could have been really difficult without their inspiration and instructions. I am really thankful to Dr. A. Lynn Abbott for helping me with computer vision methodologies and performance evaluation platform design. A regular feedback on my research work helped me a lot to keep things moving on time.

Special thanks to Seth Peery, Luke Ward and entire EGIS team for supporting and guiding me throughout my studies at Virginia Tech. Thank you to everyone, faculties, teammates who were involved with Mirror Worlds project since last year. I would like to thank my virtual environment class project group who helped me to conduct user study for collaborative object manipulation task. I am thankful to ICAT staff at Virginia Tech for providing me perfect environment to work on my thesis.

Thanks to all my friends and students at Virginia Tech for making my time here so memorable. Finally, my thesis is dedicated to my parents and family for their ceaseless love and care, which always encouraged me to overcome the most stressful moments and complete my Masters Degree.

All photos by Author.

Sachin Vasant Bharambe.

Contents

1	Introduction	1
1.1	Motivation and Overview	1
1.2	Contributions	5
1.3	Outline of thesis	6
2	Related Work	7
2.1	Human Position Detection	8
2.1.1	Background subtraction/ modelling	8
2.1.2	Histogram of oriented gradients based methods	10
2.1.3	Object detection with Convolutional Neural Networks	12
2.2	Position tracking methods	14
2.3	Remote presence application	16
2.4	Collaborative object manipulation applications	18
3	System Overview	20
3.1	Data acquisition and processing unit	21
3.1.1	Input Devices	21

3.1.2	Extracted data format and transmission process	24
3.2	Server setup	26
3.2.1	Producer to server communication	26
3.2.2	Blob manager	28
3.3	3D Models	30
3.3.1	Unity clients	30
3.3.2	X3D clients	30
3.3.3	Data reception on client side	31
3.4	Creating shared space between remote locations	32
3.5	Setup for collaborative object manipulation experiment	34
4	Computer Vision Tracking	35
4.1	Introduction	36
4.2	Object Detection	36
4.2.1	Background subtraction for human position detection	36
4.2.2	Detection with neural networks	41
4.2.3	Combine approach based on background subtraction and Neural Network	45
4.3	Human Position tracking	46
4.3.1	Data Association	47
4.3.2	Track Management	48
4.3.3	Prediction	51
4.3.4	Combined heuristics based tracking algorithm	52
4.4	Evaluation	55

4.4.1	Extracting the Ground Truth Data	56
4.4.2	Detection Measurement	59
4.5	Results	60
4.5.1	Scenario I	62
4.5.2	Scenario II	64
4.5.3	Scenario III	64
4.5.4	Scaled system	65
5	Collaborative Object Manipulation	75
5.1	Introduction	75
5.2	System architecture	76
5.2.1	Hardware Module Working	77
5.2.2	Software Module Working	78
5.3	User Study Description	79
5.4	Results of the study	81
5.4.1	Quantitative results	81
6	Conclusions	87
7	Future Work	89
	Bibliography	90

List of Figures

1.1	Detected blobs with bounding boxes	3
1.2	Mapping the real world positional data as avatar in virtual model	4
1.3	Telepresence application of Mirror World project [4]	4
1.4	Two users collaboratively manipulating objects in shared environment	5
2.1	Overview of feature extraction and object detection chain suing HOG [8]	10
2.2	Feed forward neural network	13
3.1	System Overview of Mirror Worlds project	21
3.2	Frame rate analysis for different resolution	22
3.3	Architecture of the collaborative system	23
3.4	Camera Configuration Data format	24
3.5	Blob Data Structure	25
3.6	Data Processing Unit	25
3.7	Server blob structure	29
3.8	Remote presence applicaiton	33
4.1	Flowchart of detection and tracking system	36

4.2	Combine small contours flowchart	42
4.3	Combine workflow for detection in MirrorWorld	46
4.4	47
4.5	Track object structure	48
4.6	Track assignment flowchart	49
4.7	Track management flowchart	50
4.8	Heuristics based on Spawn area, region marked with blue rectangles	53
4.9	Heuristics based track management	54
4.10	VATIC video annotation flowchart for our system	57
4.11	VATIC annotation result structure	58
4.12	Comparison of 3 approaches - Frame Rate vs Time	61
4.13	SFDA vs Spatial Overlap Ratio threshold (<i>Ovlap_thresh</i>)	63
4.14	Overview of the entire Moss Art Center 3D model with avatars displaying real-time position of people present in the building	65
5.1	User interface for object modelling task	77
5.2	User study without Oculus	80
5.3	User study with Oculus	80
5.4	Correlation between Purdue Spatial Visualization Test [9] Score and Overall rating by User $R = 0.587$	82
5.5	Correlation between users experience with video games and overall rating by user $R = 0.148$	82
5.6	Correlation between users experience with 3D software and overall rating by user $R = 0.218$	83

5.7	Correlation between users experience with 3D interaction devices and overall rating by user $R = 0.086$	83
5.8	Evaluation of Leap Motion Controller	84
5.9	Evaluation of Oculus	84
5.10	Evaluation of presence of physics	85
5.11	Evaluation of effect of shadows	85

List of Tables

4.1	Confusion Matrix	61
4.2	Frames for different scenarios	66
4.3	Experimental results for scenario I	68
4.4	Evaluation results for Scenario I	69
4.5	Experimental result for scenario II	70
4.6	Evaluation Result for Scenario II	71
4.7	Experimental result for scenario III	72
4.8	Evaluation Result for Moss Art Center video sequence - Table.4.7, Room I .	73
4.9	Evaluation Result for Torgersen Hall video sequence - Table. (4.7), Room II	74
5.1	Performance analysis for tasks without Oculus	86
5.2	Performance analysis for tasks without Oculus	86

Chapter 1

Introduction

1.1 Motivation and Overview

Presence has been defined by Witmer and Singer (1998) as “the subjective experience of being in one place or environment, even when one is physically situated in another.” This definition explains the concept of remote presence through a virtual environment. The level of presence can be extended to replace real face-to-face meetings with equally satisfying distributed meetings [10] [11].

The conventional method for 2D telepresence is video conferencing, which has been significantly improved over last few years with lower cost for enhanced network connectivity, good quality camera sensors, smart devices, IoT devices, etc. The experience of video quality and audio effect through these media has improved significantly. However, these approaches have several drawbacks. Typically, applications like Skype, NetMeeting, Hangout or any other 2D video conferencing application, present the remotely connected users in separate windows or overlaid with content. This method provides low level of presence to users. The immersiveness experience and collaboration productivity can be improved by conveying gesture information and eye contact ([12], [13]), which are not possible with conventional video conferencing technologies.

DeFanti *et al.* [14] proposed a system where the geographically distributed users' positions were mapped to digital avatars in a shared virtual reality (VR) environment. By taking advantages of recent technologies like Google Glass, Atheer AiR Glasses and Microsoft HoloLens, researchers have developed a platform where the digital representations can be rendered over the real world to create a mixed reality experience [15].

Most systems [16] are only evaluated in a laboratory setting with limited users and applications. The situation becomes more complex and challenging for scaling it in arbitrary environments such as classrooms, educational buildings or offices. Some of the challenges are synchronization across data received through multiple devices, low-fidelity, low frame rate, network delays due to large data packets, changes in environmental conditions and scalability of the software. Also, installing multiple sensors for extracting position and gesture information increases cost and complexity, and reduces the scalability.

The work done in this thesis was part of the Mirror World [4] project. The infrastructure for the project was funded by National Science Foundation (NSF). This infrastructure enables researchers to investigate a wide variety of topics such as parallels between behavior in real and virtual spaces, the effects of fidelity in virtual environments on performance in physical environments, crowd model simulation and evaluation, the relationship between behavior and affect in a public venue, and computational models of energy-efficient buildings. Virtual models of Moss Art Center and Torgersen Hall were developed by students from engineering and architecture departments. In this work, a platform for real-time human position detection and tracking was developed based on video feeds received from omnidirectional cameras embedded into the ceiling. The algorithm designed for this system was a combined approach of using detection results from background subtraction [5] and neural networks [6][17]. The detection results are also called 'blobs' (Fig. 1.1), which is a group of connected foreground pixels in an image that share some common property (e.g. color, texture, etc.). The rectangle enclosing the blob is known as a bounding box [18]. Detection results were filtered based on spatial information about the physical space using heuristics. The extracted positional data is sent to the central Fusality server [19] using json packet over TCP/IP. As described by Polys *et al.* [19], the Fusality server translates the location

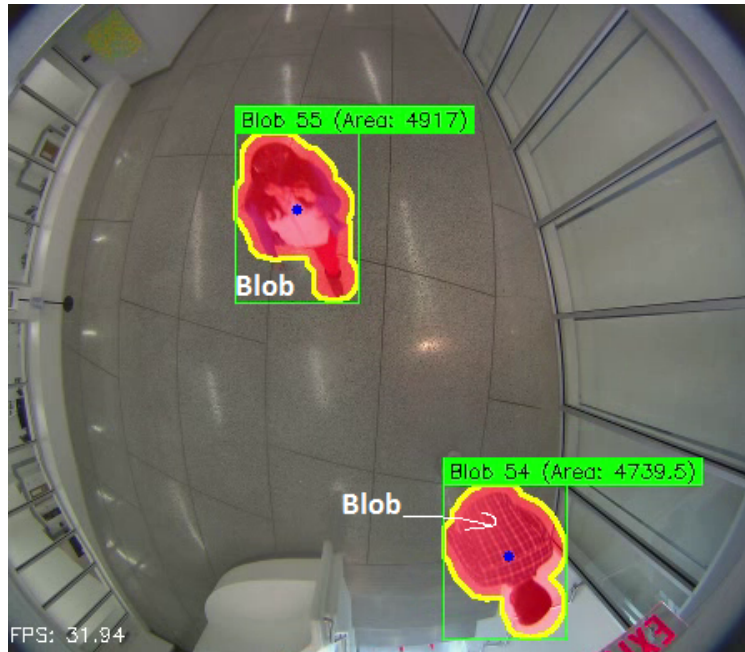


Figure 1.1: Detected blobs with bounding boxes

information from camera coordinate system to global coordinate system of virtual space. The translated data is broadcasted to all 3D Unity/ X3D clients. The location information received on clients is represented as an avatar in a model. This Model is rendered on tablets mounted on side walls displaying the mirrored view of the physical space near to the tablet. The setup gives a real-time reflection of people present in the building as an avatar into model(Fig. 1.2).

The proposed platform enabled participants from remote locations to be represented as different avatars (Fig. 1.3, red avatars) within the shared virtual environment to differentiate from local participants (Fig. 1.3, blue avatars). The collaboration platform between people from different geographic locations was developed using additional Leap Motion Controller 1.4. A user study was conducted to to evaluate the effect of fidelity in performing collaborative object manipulation tasks.



(a) Reflection with TV as mirror (b) Reflection with tablet on side wall as mirror

Figure 1.2: Mapping the real world positional data as avatar in virtual model

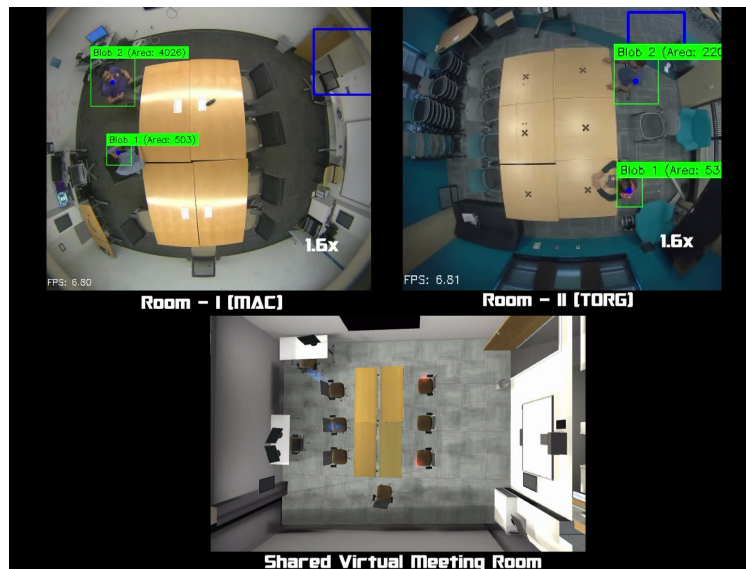


Figure 1.3: Telepresence application of Mirror World project [4]



Figure 1.4: Two users collaboratively manipulating objects in shared environment

1.2 Contributions

The main contributions of the thesis are as follows:

1. Designed an algorithm to detect and track people moving in physical using a combined approach of background subtraction [5] and neural network[6][17] with heuristics based on spatial information about the physical space.
2. Scaled the detection and tracking system to support multiple omni-directional cameras (28 across Moss Art Center and Torgersen Hall).
3. Developed a shared virtual place by mapping people from different geographical location to the shared environment.
4. Created an application using Leap Motion Controller to perform collaborative object manipulation tasks in a shared environment.
5. Conducted a user study to evaluate the effect of fidelity for collaborative object manipulation tasks.

1.3 Outline of thesis

- Chapter 2 discusses existing work in blob detection, blob tracking, remote interactions, and collaboration in shared virtual environments.
- Chapter 3 gives an overview about the Mirror World project's system architecture and design overview of application for remote presence, collaborative object manipulation
- Chapter 4 covers details of the combined approach using background subtraction[20][5] and neural networks[6][17]. It also explains the modified data association algorithm [21] based on heuristics. The evaluation subsection gives details about a platform created to evaluate the performance with Frame Detection Accuracy (*FDA*) [7] followed by results.
- Chapter 5 describes an application designed for collaborative object manipulation task followed by findings from the user study.
- Chapter 6 summarizes results obtained from user study findings and proposed algorithm.
- Chapter 7 discusses some of the future work related to improvement of design of the algorithm and creating more immersive application using additional sensors.

Chapter 2

Related Work

In this section, we will go through the related work done in human position detection and tracking algorithms, and design overview of applications developed in literature for remote presence and collaborative object manipulation in virtual environment. It will be structured as follows:

- Section 2.1, will overview human position detection techniques, such as background subtraction, HOG detectors [8] and R-CNN [22] based detectors.
- Section 2.2 will discuss tracking methods using mainly focused on tracking-by-detection and data association technique.
- Section 2.3, gives an overview about the other parallel systems based on “Mirror World” [4] idea which map the real world event to virtual environment.
- Section 2.4, presents relevant the work done in the literature for collaborative object manipulation in virtual reality using different input devices.

2.1 Human Position Detection

The detection techniques will try to differentiate background from the foreground objects. Background can be considered as a relatively static environment where environment changes gradually over time due to illumination changes or wind etc. On the other hand, foreground objects are treated as moving objects in the scene. There are 3 different approaches to separate out the foreground objects from the background.

- Background subtraction/ modelling: In this method, a background model is created to represent the environment. Each new input frame is compared to the background model and if pixels are significantly different than the corresponding pixels in model then it is claimed as foreground object.
- Histogram of oriented gradients with SVM classifiers: This approach uses the intensity gradients to determine the shape of object. It decomposes entire image into different cells and for each cell histogram gradients are computed in horizontal and vertical direction. Once all the features are identified then SVM, adaboost classifiers are used for identifying humans.
- Object detection with R-CNN [22]: The detection method is distributed in 3 primary modules. First, generating independent category region proposals. Second, large convolutional neural network that extracts fixed length feature vector from each region. Third, class specific linear SVMs to detect the object.

2.1.1 Background subtraction/ modelling

The conventional background subtraction method relies on single image reference as background. The foreground objects are determined by (2.1),

$$I_{foreground} = |I_{input} - I_{background}| > T \quad (2.1)$$

where, $I_{foreground}$ is foreground object image, I_{input} is input frame, $I_{background}$ is background image, and T is difference threshold. In this approach each input frame pixel is subtracted

from the background image pixel value. If this difference is greater than some threshold T then it is claimed as foreground pixel. This kind of approach works when the background is not changing with respect to time.

However, this method creates lot of noise in the measurement. Hence, instead of considering background reference as a single image, researchers developed a representation of the scene called the background model. The deviation of each new frame is calculated from the model to estimate the foreground objects. The frame differencing of temporally adjacent frames is studied since late 70s [23]. To adopt the gradual changes in background, Wren *et al.* in [24] proposed model for the color of each pixel of image. This model tries to fit Gaussian probability density function (pdf) on last n pixel's values. A Gaussian distribution $N(\mu_{s,t}, \Sigma_{s,t})$ where $\mu_{s,t}$ and $\Sigma_{s,t}$ represent average background color and covariance matrix at pixel s at time t respectively. Pixel intensity distributions are not uniform across entire frame. Hence, single Gaussian is not good for changing background [25]. Grimson and Stauffer [26] proposed the multimodal statistical models for representing the environment as a probabilistic view to determine whether the object belongs to foreground or background. In this approach, they are using Mixture of Gaussians (MOGs) to each and every pixel in background. Every pixel in the input frame is compared against the background model by finding the deviation with every Gaussian in the background model until a matching Gaussian is found. The probability of occurrence of a color at given pixel s is given by: $P(I_{s,t}) = \sum_{i=1}^K \omega_{i,s,t} \cdot N(\mu_{i,s,t}, \Sigma_{i,s,t})$ where $N(\mu_{i,s,t}, \Sigma_{i,s,t})$ is the i^{th} Gaussian model and $\omega_{i,s,t}$ its weight. Stauffer and Grimson suggested that covariance matrix $\Sigma_{i,s,t}$ can be assumed to be diagonal matrix given by $\Sigma = \sigma^2 Id$. Parameters for the matched component which is the nearest Gaussian for which $I_{s,t}$ is within 2.5 standard deviations of its means are updated and parameters of unmatched distributions remain the same with just weight update to achieve decay.

Elfamall and Davis [27], proposed a kernel density based approach to create background model per pixel. In the matching process the pixel is not just matched with the corresponding pixel in the background model but also matched to neighborhood pixels to eliminate the distortions due to camera movements or small jitters in background. In 1999, Toyama *et al.* [28], proposed a method based on not just pixel based background model but it involves

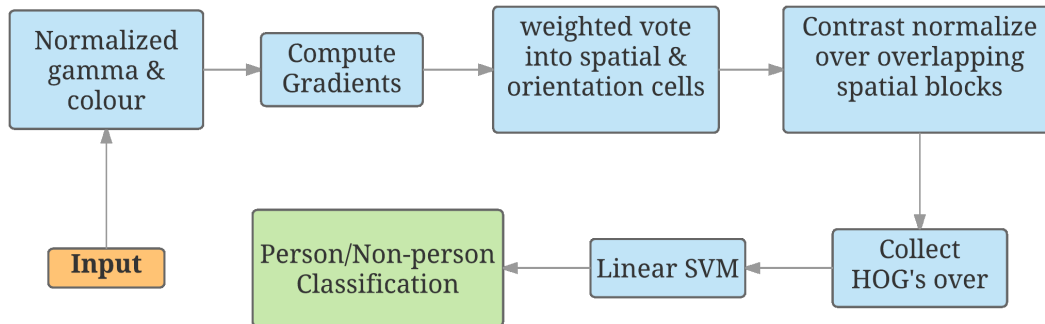


Figure 2.1: Overview of feature extraction and object detection chain using HOG [8]

the complete frame and region information. At pixel level they used Wiener filter to make probabilistic predictions, region based information is collected by finding the homogeneous colors in the images. If most of the pixel in the background are changed then it is assumed that environment is changed and previous model reinitialized based on new values.

2.1.2 Histogram of oriented gradients based methods

These methods are based on selecting the feature sets for human detection based on locally normalized Histogram of Oriented Gradient (HOG) descriptor. Dalal and Bill proposed that HOGs feature sets performs better relative to other feature sets including wavelets [29][30]. This method uses the normalized histograms of image gradient orientations. There are similar approaches for human detection task based on these features [31][32] [33]. The work-flow for Dalal and Bill method can be represented in Fig. (2.1).

In this approach, entire input image is divided in small grid cells and for each such cell histogram of gradient directions or edge orientations is computed. The combination of all these histograms forms a representation and to enhance the quality of this representation is contrasted and normalized to local responses before it can be used further. This normalized descriptor is known as Histogram of Oriented Gradient (HOG).

Felzenszwalb and Girshick, proposed object detection with discriminatively trained part-

based models (DPM) [34]. In their approach, they used histogram of gradient (HOG) [8] features variants, where the feature size is reduced such that there is no loss in performance. They used PCA to convert the higher dimensional features into lower dimensions [35]. Using this approach 36-dimensional feature vector represented in [8] reduced to 13-dimensional feature set that captures essentially the same information. This low dimensional data is then augmented to 31-dimensional feature set to include contrast sensitive and contrast insensitive information. The lower dimensional feature sets lead to speed up the detection process. These feature maps are used with linear filters F . This filter F is a rectangular template ($w \times h$) defined by an array of d -dimensional weight vectors. The response of the filter F at position (x, y) in feature map G is the “dot product” of the filter and sub window of feature map G , with top left corner at (x, y) :

$$score(p) = \sum_{x', y'} F[x', y'] \cdot G[x + x', y + y'] \quad (2.2)$$

where $score(p)$ is score received at position (x, y) . This score is defined at different positions by using feature pyramid. If H is feature pyramid and $p = (x, y, l)$ specifies position (x, y) in the l th level of pyramid. Precisely score of F at p is computed by (2.3),

$$F' \cdot \phi(H, p, w, h) \quad (2.3)$$

where F' is concatenating the weight vectors in F , $\phi(H, p, x, y)$ is obtained by concatenating feature vectors in $w \times h$ subwindow of H with top left corner at p . The idea of deformable parts model consist of one root filter F_0 along with part filters (p_1, p_2, \dots, p_n) that covers smaller parts of the object generally with higher resolution feature space. Hence complete n parts deformable model can be represented by an $(n + 2)$ -tuple (F_0, P_1, \dots, P_n) . Each part is defined by (F_i, v_i, d_i) where F_i is filter for i^{th} part, v_i is vector denoting the relative position from the root filter location, d_i vector denotes the deformation cost for each possible placement of part relative to root position. To detect the object in an image, overall score for each cell location is computed according to best possible placements of parts:

$$score(p_0) = \max_{p_1, \dots, p_n} score(p_1, \dots, p_n) \quad (2.4)$$

where, score is given by the difference between scores of each filter at their respective locations and deformation cost that depends on the relative position of each part with respect to root

location with bias given by equation (2.5),

$$score(p_0, ..., p_n) = \sum_{i=0}^n F'_i \cdot \phi(H, p_i, w, h) - \sum_{i=1}^n d_i \cdot \phi_d(dx_i, dy_i) + b \quad (2.5)$$

where,

$$(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i) \quad (2.6)$$

gives displacement of i^{th} part relative to its root position. Hence to locate the object location in whole image, the sliding window approach is used to determine best location for root position. The time complexity of the method is $O(nk)$ after filter responses are computed, where n is number of parts in the model and k is total number of locations in the feature pyramid.

2.1.3 Object detection with Convolutional Neural Networks

The design of convolutional neural networks is typically the extension of the feed forward neural networks with multiple layers. The network consists of input layer, hidden layers, and an output layer. The nodes in these hidden layers and output layers acts as neurons with a suitable activation function. The output of these neurons can be given as (2.7),

$$output = f(\vec{x}, \vec{\theta}) \quad (2.7)$$

where, the f in Fig.2.2 represents the activation function, \vec{x} is input vector and $\vec{\theta}$ is weights vector. The complete network can be represented by Fig.2.2. The activation function can be represented as binary step function (Perceptron), log-sigmoid, linear, tanH etc. For example, if the activation function used is sigmoid then can be computed by [36],

$$f(b + \sum_i w_i x_i) = \frac{1}{1 + e^{-(b + \sum_i w_i x_i)}} \quad (2.8)$$

In the network, Fig.2.2 consider the input as an image. In this case each pixel can be treated as separate input. Hence, the input size scales quadratically with the resolution image [36]. The solution of this problem is to connect each hidden unit to small patch of the input image

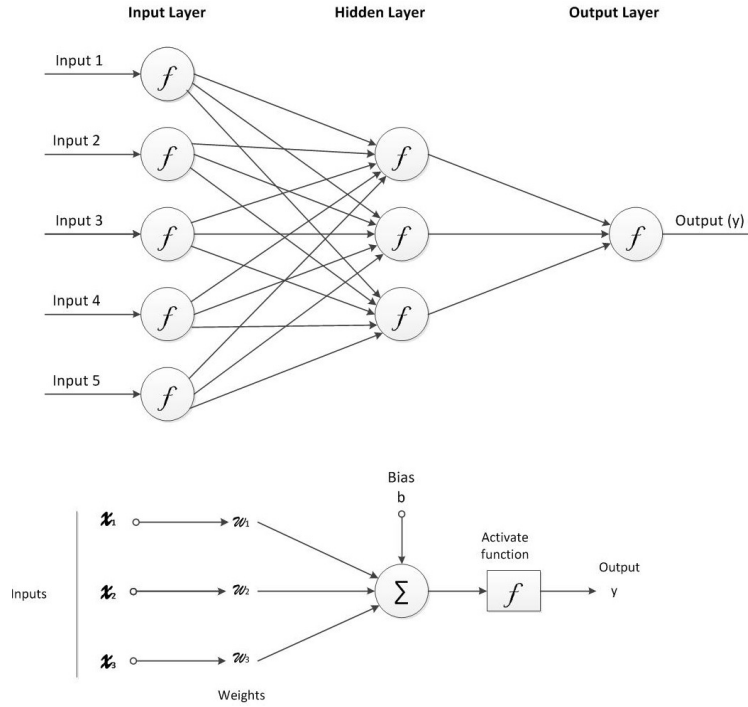


Figure 2.2: Feed forward neural network

and share the weight across hidden units. This will help to scale the neural network solution to images with larger resolution. This concept is known as convolutional neural network [37].

Krizhevsky *et al.* [38] proposed the use of convolutional neural networks for object classification. The training was done on ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[38][39]. The network designed for this task was based on eight layers which includes five convolutional and three fully connected layers. They used rectified linear units (ReLUs) as activation functions for neurons. The extension of classification results on ImageNet [40] is proposed by Girshick *et al.* [22] 2014, which was based on region based convolutional neural networks (R-CNN) for object detection on PASCAL VOC [41].

The detection requires localizing the object within image, while classification task does not aim for localization. The localization task can be done by considering as regression problem, which was proposed by Szegedu *et al.* [42] whereas in this paper [22] the localization problem is tackled by operating within the recognition using regions paradigm. The requirement of

large CNN is to have a labeled data source. The problem of insufficiency of the data is handled by adopting unsupervised pre-training, followed by supervised fine tuning [43].

Region based convolutional neural network (R-CNN) detection algorithm [22] is based on 3 modules. The first module is region proposals. Region proposals in this detector are based on selective search [44] to enable a controlled comparison with prior detection work [44] [45]. The second module is about feature extraction where Girshick *et al.* [22] used Caffe [46] implementation of CNN described by Krizhevsky *et al.* [40]. They extracted 4096 dimensions feature vector from each region proposal. These extracted features are initially passed through the network to get good quality features and using class specific SVM classifier score is computed for each feature per class. Then a greedy non-maximum suppression is applied to these scores spread over the regions. If the intersection-over-union (IOU) overlap with higher scoring selected region is greater than learned threshold then that region is rejected.

2.2 Position tracking methods

Human position tracking involves not only localizing human presence in the scene but also involves detecting same target across multiple frames with consistent detection identity. The problem of tracking multiple targets through a complex scene involves challenges such as object location uncertainty, clutter, measurement noise, changing background and significant occlusions. Generally, tracking methods need an input from object detection (section 2.1) process or input from motion model. Based on the input type tracking can be categorized in 2 parts:

- First is tracking by detections, where system assumes a reliable detector (robust detection algorithm) and also assumes that these detections are well spaced in images. In order to resolve the difficulties involved in tracking task, researchers have preferred *tracking-by-detection* approaches [47] [48] [49] [50].
- Second is tracking by flow, where the system assumes unknown appearance but do-

main, parametric flow models are known [51] [52] [53]. These methods select the best parameters from the flow models that aligns object in one frame to object in next multiple frames.

In this work, tracking based on detection approach is used to track human position. The tracking algorithms based on data association are based on linking detection responses to generated trajectories (Tracks) by global optimization based on position, size, and appearance similarity [54], [55], [56], [57], [58], [59].

Breitenstein *et al.* in 2009 proposed tracking-by-detection using detector confidence particle filter. The key contribution of this paper was to present an algorithm to automatically detect and track multiple targets in complex scenes using monocular cameras. Primary contributions of this paper are: Design of probabilistic tracking-by-detection in Particle Filtering framework, use the confidence of filter and integrate it with the observation model, and solve unrealistic detections using online trained classifiers along with input data. The optimal data association can be achieved using Hungarian algorithm [60] but authors discovered that an greedy algorithm achieves similar result at lower cost. In this algorithm, score matrix S is computed for each pair (tr, d) of tracker tr and detection d . All associated detections with matching score above threshold are used, ensuring that a selected detection actually is a good match to a target where the score is estimated based on equation (2.9). The greedy algorithm used in this method can be described in Algorithm. 1,

$$S(tr, d) = g(tr, d) \cdot \left(c_{tr}(d) + \alpha \cdot \sum_{p \in tr}^N p_N(d - p) \right) \quad (2.9)$$

where, $p_N(d - p) \sim N(pos_d - pos_p; 0, \sigma^2)$ denotes the normal distribution evaluated for the distance between the position of detection d and a particle p , and $g(tr, d)$ is a gating function. This gating function depends on 2 normal distribution. First that measures the agreement between the bounding box height of target and detection and follows the intuition that fast-moving objects cannot change their course abruptly because of inertia. Hence this term

Algorithm 1 Greedy data association

 T : set of all trackers D : set of all detections $S(tr, d)$: scores for each tracker-detection pair, Equation (2.9) $A(tr, d) = 0$: final associations of detection d to tracker tr **Require:** $\forall tr \in T : \sum_i A(tr, i) \leq 1$ **Require:** $\forall d \in D : \sum_j A(j, d) \leq 1$ **while** $T \neq \phi \wedge D \neq \phi$ **do** $(tr^*, d^*) = \arg \max_{tr \in T, d \in D} S(tr, d)$ **if** $S(tr^*, d^*) \geq \tau$ **then** $A(tr^*, d^*) = 1$ **end if** $T = \{T \setminus tr^*\}$ $D = \{D \setminus d^*\}$ **end while**

depends on the velocity of target. The gating function $g(tr, d)$ is given by (2.10),

$$\begin{aligned}
g(tr, d) &= p(size_d|tr)p(pos_d|tr) \\
&= \begin{cases} p_N\left(\frac{size_{tr}-size_d}{size_{tr}}\right) \cdot p_N(|d-tr|), & \text{if } |v_{tr}| < \tau_v \\ p_N\left(\frac{size_{tr}-size_d}{size_{tr}}\right) \cdot p_N(dist(d, v_{tr})), & \text{otherwise} \end{cases} \quad (2.10)
\end{aligned}$$

where, $v_{tr} = (u, v)$ is given by position of tracker and direction component of the velocity.

2.3 Remote presence application

In the literature many authors define the presence in variety of ways. Seminal authors Short, Williams & Christie (1976) defined it as “the degree of salience of the other person in a mediated communication, and the consequent salience of their interpersonal interactions” (p. 65). In general terms, it can summarized as a feeling that someone exist around for an interaction. Social presence is affected by variety of other factors that contribute to feelings

of social presence. Some of the factors could be “immediacy”, initially said by Wiener & Mehrabian (1968), and “intimacy”, as was described by Argyle & Dean (1965) are two factors that define a presence in literature.

The presence can be experienced through virtual environment where the interaction are not just limited to specific physical location and specific virtual worlds but it also enables people located elsewhere to connect and represent themselves in the shared virtual world as avatars. Hence participant connecting from different location become part of the virtually rendered spaces that more closely emulated to the physical spaces.

Gelernter [61] proposed that Mirror worlds are distinguished from other virtual representations of the real world by providing real-time mapping from real-world objects to software equivalents. It is also described as as dual reality or cross reality as realized in the mirror world built of the MIT Media Lab [62]. Mix reality systems can be designed using Physical/digital integration systems such as Phidgets and Arduino, which will act as a source of information, sensor/actuator for mixed reality [63].

Back Maribeth *et al.* [1] developed a real world mixed reality system for industrial collaboration and control. The industry was a start up whose focus was creating “high-tech chocolate” by applying new technologies. The purpose of this design of mixed reality environment was not just replicating the factory into virtual world, but to create an electronic infrastructure that can captured data from many of the companys processes, and visualize all this information for in virtual world. The access level is decided on the people designation and role. The system was using multi-camera video streams input from network cameras and text data as input source. The data recorded from sensors and machines logs were parsed via XML. This parsed data is distributed into the Virtual Factory world as floating text against a transparent color cloud, called a “Data Spot.” These Data Spots are animated to visualize sensor data such as hot water temperature, chocolate temperature, or machine state. In addition to this, the virtual tour for the factory is designed such that avatar in the model gives users a complete your of factory by going to each machine.

Kimber *et al.* in 2011, proposed a mirror world system at FXPAL which mirrors a physical space of the research lab into cyberspace to provide people with augmented awareness of

space. They even claim that in this system people present in one part of a building can get a sense of the activities in the rest of the building, who is present in their office. The idea behind creating this mirrored space as ‘community level perceptual system’ to support activities and awareness of the people involved [61]. The key elements of their system was models, maps and representations of the layout, sensor network and viewer applications. The system was dependent on 20 surveillance cameras which were used to track the people’s movement in the environment. They have mentioned that the accuracy of tracking by these plain camera sensor was not good. They used additional two TYZX stereo cameras [64] to track the people movements accurately. They also integrated the MyUnity awareness system which collects the data from bluetooth devices, keyboard activity, calendars and cameras across 20 office regions to count the number of people present in those areas. The viewer application designed for this project is compatible with portable devices, windows application or can even run on web page.

2.4 Collaborative object manipulation applications

Extensive research in this field has been carried out with the intention of evaluating 3D input devices in terms of 3D interaction techniques and its relation to user performance. As a result of constant development of interaction devices and rendering systems, research in this field is still a common practice.

A study by Joann *et al.* [65] on Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment provides a good evaluation of 3D pointing tasks using Leap Motion sensor to support 3D object manipulation. In this paper three controlled experiments were performed in the study, exposing test subjects to pointing task evaluations and object deformation, measuring the time taken to perform mesh extrusion and object translation. Qualitative data was gathered using the System Usability Scale questionnaire. Their data reveals a strong correlation between input device and performance time suggesting a dominance of the Leap Motion gestural interface over mouse interactions. Based on their results from the experiments, they concluded that presented 3D input device, leap motion outper-

formed mouse interaction only in single target situations, showing that 3D translation is less cumbersome when the z axis is provided as input based on real-life movement mapping.

Raj, Creem-Regehr *et al.* [66] proposed Kinect Based 3D Object Manipulation on a Desktop Display. The system presents a controlled experimental evaluation of the use of Microsoft Kinect to support a 3D object manipulation task. Users were asked to match the orientation of objects with a manipulation interface that displayed either a self-avatar hand and arm or a sphere, both corresponding to users arm gestures and wrist rotation. Their results showed that while there was no overall difference in performance between the different avatars but there were clear differences in the two visual display conditions as a function of gender and video-game experience.

Research performed by Scheggi *et al.* [67], on Touch the virtual reality: using the Leap Motion controller for hand tracking and wearable tactile devices for immersive haptic rendering, the authors of this research presented a novel haptic system for immersive virtual reality interaction. In their research users were asked to wear five tactile devices on one hand and interact with different virtual objects while their hand pose was tracked using the Leap Motion controller. A virtual hand mimicked the users hand pose. Every time the hand came in contact with a virtual object, the tactile devices applied a suitable amount of force to the users fingertips, providing them with the compelling sensation of touching the virtual environment.

Otmar Hilliges *et al.*, proposed system on HoloDesk: Direct 3D Interactions with a Situated See-Through Display [68]. The authors presented a novel device called as HoloDesk which has the ability to capture user gestures which would in turn be translated to actual object manipulation that could be viewed in a see-through screen incorporated as a part of the device. The system employs Kinect sensors to record user gestures. The paper then describes various techniques and gestures involved in that process.

Chapter 3

System Overview

This chapter will give explain the system architecture of Mirror Worlds project followed by design overview of application for remote presence and collaboration. It will be structured as follows:

- Section 3.1, explains data processing unit and its functionality.
- Section 3.2, will discuss different aspects of the Fusality server's [19] and it's role in Mirror World project.
- Section 3.3, gives functional overview of 3D clients (unity /X3D) as virtual environments.
- Section 3.4, describes the tele-presence application through Mirror Worlds platform.
- Section 3.5, gives details about the setup for collaborative object manipulation task as an extension of Mirror Worlds using additional sensor.

The Fig. 3.1 shows the system overview of the Mirror Worlds project.

The input was taken from network fish-eye cameras and Leap Motion controllers. The data acquisition and processing was done on local client using OpenCV (for computer vision) and Unity (for Leap gesture recognition). The outputs of data processing units i.e. spatial

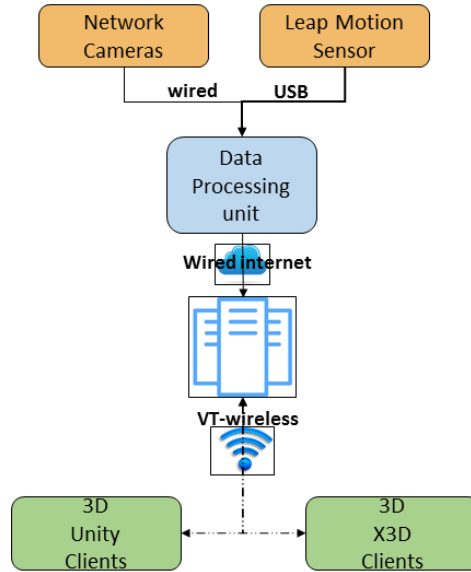


Figure 3.1: System Overview of Mirror Worlds project

co-ordinates for blob tracking and recognized gestural information was sent through central Fusality server [19] over network to the 3D Unity/ X3D clients (mirrors), which render virtual 3D models of the physical world.

3.1 Data acquisition and processing unit

This unit has an important role in building a base setup for Mirror Worlds project. The processing unit acquires the video feed from network Omni-directional cameras.

3.1.1 Input Devices

3.1.1.1 Network cameras

The network cameras (On-Cam / Vivotek), can transmit the 3 simultaneous H.264 and mjpeg video streams, with resolutions varying from 528 480 (1/4 MP) to 2144 1944 (4MP). This

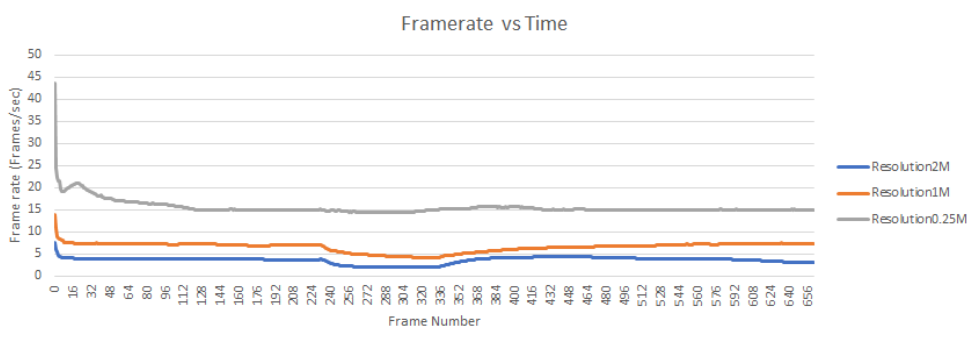


Figure 3.2: Frame rate analysis for different resolution

transmission occurs at maximum 15 fps with fish-eye view and 30FPS with normal video view. It is clear from the Fig. 3.2 that as video resolution increases the processing speed of the system reduces significantly. In this system, omni-directional camera was used to cover a larger physical area. Transmitted image resolution was set to 1056 960 (1 Megapixel) to balance the processing speed and information details. All network cameras were configured to work in private network. The configuration setup was done through the IW2 software for Vivotek cameras [69] and Camera Configuration tool [70] for On-Cam cameras. The detailed procedure is in listed in [71]. About 28 of these network cameras are installed in the Moss Art Center, and eight are installed in Torgersen Hall. As all cameras configured over private network with password, raw video feed is not available publicly. It can be only accessed via data acquisition and processing unit in the building.

3.1.1.2 Leap Motion controller

The Leap Motion controller is an usb peripheral device which can be interfaced with the PC or laptop. It is also possible to mount the device on VR headsets. In this work, Leap Motion controller V2 was used for experiments. The sensor is made up of two monochromatic IR cameras and three IR LEDs. The functional zone of the device is approximately one meter radius hemisphere from the center of the device. In the system, Leap Motion controller was interfaced with Unity software on windows platform. The gestures performed within the functional cone of sensor were used for interaction with virtual objects. The proposed

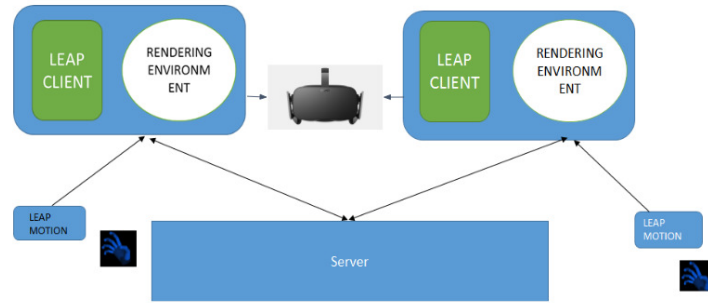


Figure 3.3: Architecture of the collaborative system

system detects pinch, drop actions and these actions are then replicated into the virtual model. Based on the actions performed by user, the virtual objects (Cubes) in this system were manipulated in shared 3D environment. The setup is shown in Fig. 3.3

3.1.1.3 Camera configuration

The data acquisition and processing unit must connect to the camera with secure access to get video feed from the camera. To correlate the extracted blob data with the actual physical space, it is required to associate the blob data with its camera id which will be unique and will be assigned at the time of installation of the device. Hence the association of this unique camera id with the blob data helps blob manager on the server to transform these coordinates to match the physical space with the virtual 3D model. Hence, while initializing each camera in the processing unit, the camera id information is collected along with its location name and cropping co-ordinates. As there are two types of cameras installed in the Moss Art Center and Torgersen hall, camera class information is included along with the Camera id because the connection url for these two different class of cameras are different. All the configuration related data is listed in the configuration file in yml format. The configuration file content is summarized in Fig. 3.4. The server url (dev.mirrorworlds.icat.vt.edu: Development Server, mw.icat.vt.edu: Production Server) and port number (9999) are used to establish a secure connection with the server. The development server is used to test the new algorithm and change in system architecture while the production server is mainly used for running stable version of application.

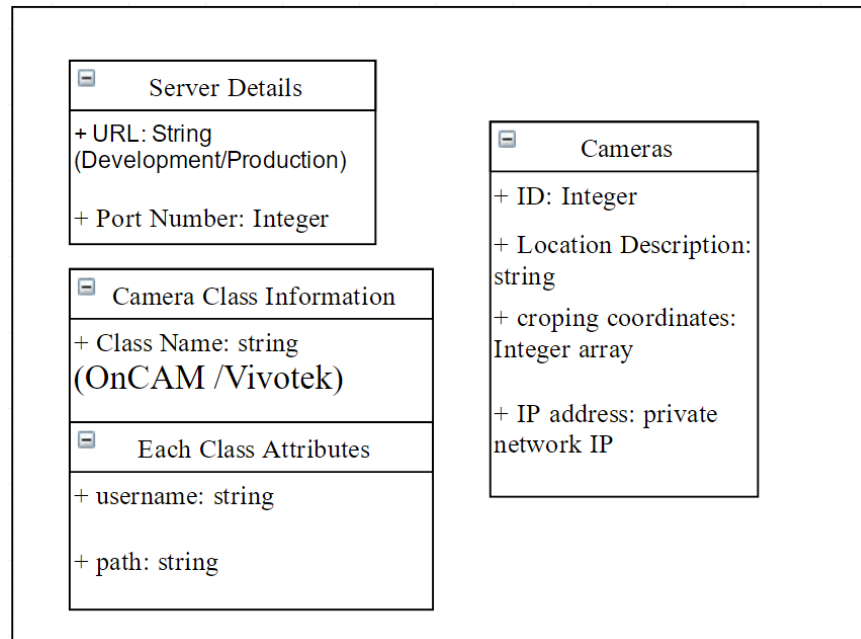


Figure 3.4: Camera Configuration Data format

3.1.2 Extracted data format and transmission process

This data processing unit acquires the raw video frames and extracts the person location information from the background using computer vision algorithms. The detected person in the input frame is called foreground object. This object is represented as contour in our system (OpenCV), based on which the area information is extracted from bounding box for detected blob. All this data is extracted and reformatted in blob structure format shown in Fig.3.5. The Camera id field for the Blob structure is obtained from the configuration file and rest of the information is obtained through detections and tracing algorithm running on the processing units. This blob data is then transmitted to server on port number 9999. The module will establish a secure tcp/ip connection with the server and all this information then reformatted in json format is sent to the server. On the overview level this whole system can be summarized in Fig. 3.6.

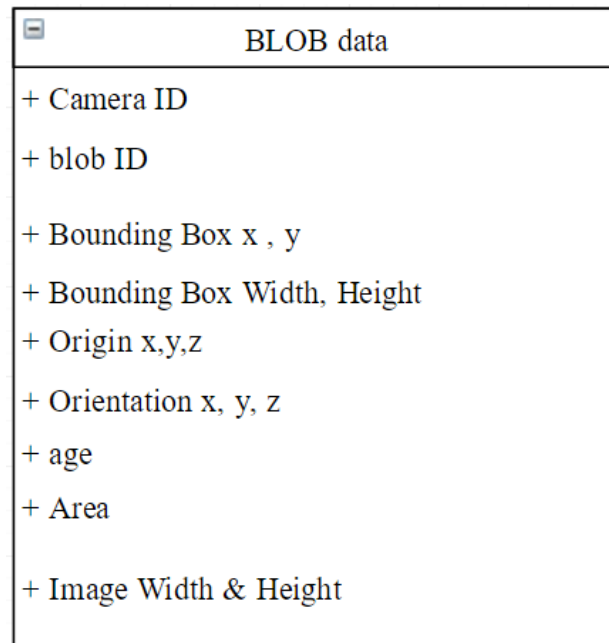


Figure 3.5: Blob Data Structure

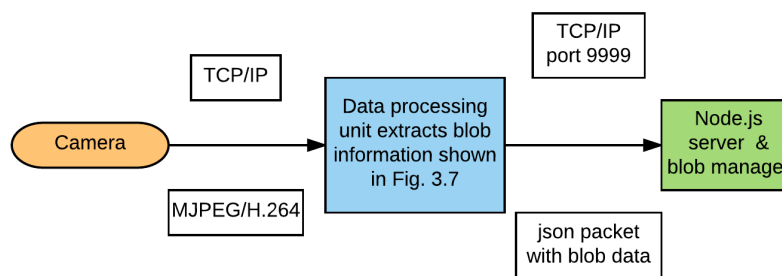


Figure 3.6: Data Processing Unit

3.2 Server setup

This project has a Fusality server [19] as a bi-directional communication medium between the data processing unit and 3D display client. The server has 2 parts as described in by Polys *et al.*,

- **Producers:** These units are generators of information to Fusality. All these are as unidirectional clients, they connect to the server to contribute information about entities such as position and motion like computer vision client for processing video data obtained from the network cameras.
- **Consumers:** They use information in Fusality to create platforms between virtual and physical spaces. Consumer are not used to provide any information to Fusality.
- **Design:** Fusality server is designed to establish a shared layer for Mirror Worlds project. It is built based on W3C api and fully-duplex tcp protocol of web sockets. The server back-end is written on top of node.js (<https://nodejs.org/>). The communication link can support 174 communication over port 80 through Socket.IO (<http://socket.io/>) library. The server receives messages from the connections it holds, which are in the form of json strings. It not only receives and transmits the information, but also runs a blob manager who plays an important role to modify the received information. This blob manager transforms the received information from the physical environment captured by producers to easily immerse into virtual environment at consumer level.

3.2.1 Producer to server communication

Server receives data from tcp/ip connections and Socket.IO library. The data received from both the methods is same. After reception of this data, server goes through validation. This validation process checks that this blob has a correct id, camera id, and age so that it can properly be distinguished from other blobs. All this information communication is in json format to match with the earlier listed blob specification Fig. 3.5.

3.2.1.1 Computer Vision data processing unit TCP module

The TCP socket connects to `http://dev.mirrorworlds.icat.vt.edu` on port 9999. This communication is one way. While sending this blob data from computer vision processing unit, after each blob the delimiterator ‘&’ is used so that each blob can be interpreted separately. Socket.io: The communication link subscribes the client as a listener so that it can receive information from other blob senders as well. To connect to the server over this socket.io link one should connect to “`dev.mirrorworlds.icat.vt.edu/ mw.icat.vt.edu`” on port 9999. To begin a connection the sender should sent a start event to the server. There are 3 types of connections that can be established with the server by following API:

```
socket.emit('start', connectionType: 'TYPE');
```

where, ‘TYPE’= {‘TWOWAY’, ‘DATASOURCE’, ‘LISTENER’}. ‘TWOWAY’ connections allow client to send and receive data from server, ‘DATASOURCE’ types clients can only send the data to server and ‘LISTENER’ clients will only listen the information broadcasted by the server.

3.2.1.2 Blob updating process

The producers connected to server with ‘DATASOURCE’ or ‘TWOWAY’ can send the blob information to server. As this data is validated by server incorrect packets will be rejected if those are not consistent with the structure shown in Fig. 3.5 . The producers update the blob data with following API:

```
socket.emit('EVENT', blob);
```

ID and age filed on the blob should match to ‘NEW’ to validate the event. The ‘update’ event is for updating previously generated blob and should have age filed as ‘OLD’, while ‘remove’ blob is to remove the preexisting blob from the world as it is no longer found by the data processing unit and will age field will be updated by ‘LOST’. The life cycle of blob should start with ‘new’ event, then ‘update’ for updating its location information and finally

‘remove’ to remove the blob from world. If the life cycle does not match with this then blob data will be rejected by the server.

3.2.2 Blob manager

Along with just reception and transmission of the information the server also performs a blob management task. The blob information sent from producers is extracted on the server and new blob structure is created on server which has very similar blob schema as described in Fig. 3.5 with some additional fields shown in Fig. 3.7. The structure in Fig. 3.7 has update and create time stamp information which can be used to log the data. This information is helpful to perform the qualitative analysis on the capture blob information. We can extract the engagement of the user with environment by evaluating the time spent by the users at different location while moving in the building, similarly it will be helpful to approximate population map in the building at any time which will be useful to discover some interesting patterns with the population flow in the building. The blob manager is part of the server. Blob manager constructor is called whenever there is send event on the server. The purpose of the blob manager to coexist with the main server is to preform CPU intensive tasks. Once this blob has been sent to the manager it is no longer the responsibility of the calling class. Blob manager will then transform each blob coordinates from the perspective of each individual camera in the physical space to a global coordinate system in virtual model that does not require each client to be knowledgeable of the specific locations and orientations of each camera, and makes the computation must happen only once instead of on every individual client. Each camera has an x, y, z, width, height, and theta associated with it. These are all determined by physically measuring the respective qualities from a predetermined origin. To aid in the addition of new cameras and more easily keep the server able to support all cameras, the data is read in from a csv file containing all the relevant information. The use of these global coordinates allows the easier mapping of movements of blobs without having to worry about the properties of each individual camera.





	Server BLOB Structure
+ Camera ID :string	
+ blob ID: string	
	Origin
+ x, y, z float	
	Orientation
+ x, y, z, theta: float	
+ Source: string	
+ updateTime: time_t	
+ Creation Time: time_t	
	Bounding Box
+ x,y: float	
+ image width,height: float	
+ width,height: float	

Figure 3.7: Server blob structure

3.3 3D Models

The Mirror World can be visualized as 3D model. There are many tools available to develop these 3D models. The common tools used for browser friendly 3D model development like Virtual Reality Modeling Language (VRML) [72], and its successor the Extensible 3D (X3D) [73] format. Similarly, the 3D game engines like unity and Maya can be used to design the models for various platforms including web, android, windows and MAC. These game engines provide excellent UI for designing the models and additionally softwares like Blender and Rhinoceros are compatible with Unity. The export option available in these software to fbx or obj, can be imported in Unity software. Hence, we developed our Moss Art Centers and Torgersen Hall models in Unity. Also, we exported these Unity models to fbx format and developed X3D compatible models which servers as online X3D clients for Mirror World.

3.3.1 Unity clients

The whole building model for Moss Art Center is then exported for Android platform to install on tablets. These tablets will act as mirrored view of the physical space and hence the corresponding camera in the model must be placed at the same location and height as the “mirror” tablets that are installed in building space. The origin of the model is very important to define global coordinate system for entire building. The blob manager on the server transforms the coordinates received from producers to incorporate them into virtual models global coordinate system. The model also has first-person camera and corresponding controllers. This is a mobile camera which can be controlled using the mouse and the arrow keys.

3.3.2 X3D clients

The X3D client are have similar model structure as it is present in Unity clients. We export Unity models to .fbx format which can be imported by Blender and Rhinoceros and after some minor modification it is then exported to X3D scene graph. There are ways to combine

the openness of X3D with the ubiquity of web technologies X3DOM [74]. X3DOM combines the X3D scene graph format with HTML Document Object Model [75]. This is done by leveraging WebGL, an emerging standard for embedding 3D content into webpages. WebGL is currently supported by development versions of all major browsers except for Internet Explorer. These include Opera, Firefox, Google Chrome, and Safari. Consequently, it will soon be available to many web users without the use of plugins at all. Hence these clients are open platform to involve into the Virtual environment and interact with real people present in the building into a virtual model.

3.3.3 Data reception on client side

In all the unity clients SocketIO library is used by communicate with the server. A SocketIO component is created in the model and attached to networking script. Using the instance of this component, the web-socket connection is established to the development server (dev.mirrorworlds.icat.vt.edu) or production server (mw.icat.vt.edu) on port 8888. The server is broadcasting all the blob data on this port. After obtaining the SocketIO component instance, “instanceSocketIO”, different callback functions are associated with different SocketIOEvent object. The callback function is called by,

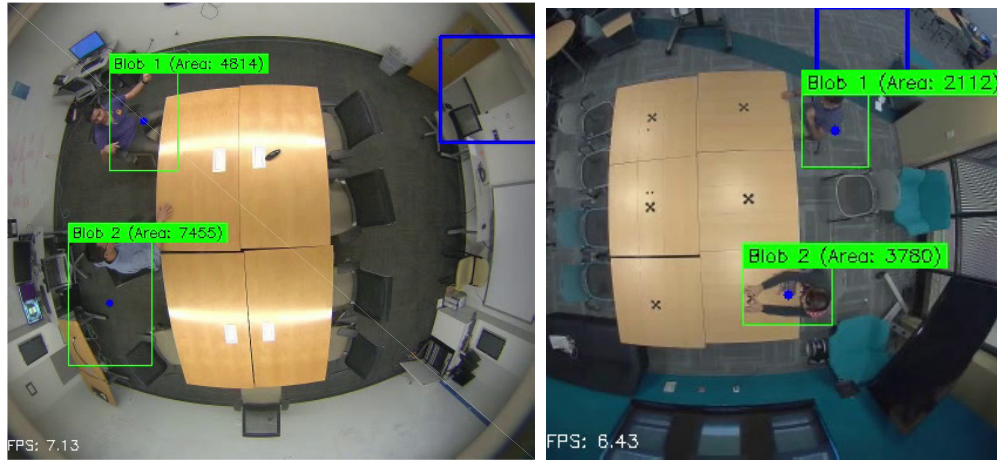
instanceSocketIO.on('EVENT_OBJECT', function f);

where function f, is different for all the possible ‘EVENT_OBJECTS’, and ‘EVENT_OBJECT’ = { ‘connect’, ‘newBlob’, ‘updateBlob’, ‘removeBlob’, ‘disconnect’}. For our project, we decided these 5 events to create, destroy and update the new avatars in the virtual space. The ‘newBlob’ callback is called whenever the new blob is detected by producers (OpenCV / MATLAB client). A signal is given to create a blob, new GameObject in the virtual model. This signal instantiates clone of prefab avatar associated with “personMarker” GameObject. The ‘updateBlob’ is to update the motion of existing avatar object, ‘removeBlob’ is to delete the existing avatar from the virtual model once the age for the blob is marked as ‘LOST’ by the producer. These call callback functions have access to the SocketIOEvent object, which is passed as an argument to the function by SocketIO library. For each blob a separate Sock-

etIOEvent is generated and “data” field in this object is extracted which represents a json object containing relevant information about the event such as camera id, blob id, and blob position. This received information is stored in dictionary object as key-value pair where value is the information about the blob and key is a unique integer identifier for the blob, which is a hash computed by combining the blob id and the camera id of the event. The hash is computed by doing a left bit shift on the camera id and bitwise ORing that value with the blob id. This hash operation generates a unique identifier which allows for roughly 1,000 different camera ids and over 4,000,000 blobs for each camera. We decided this hash function as the preferred option because it is less expensive and either limit is doubtful in the near future for this project. A simple solution for future expansion would be to use a long for the key to allow more bits for both values.

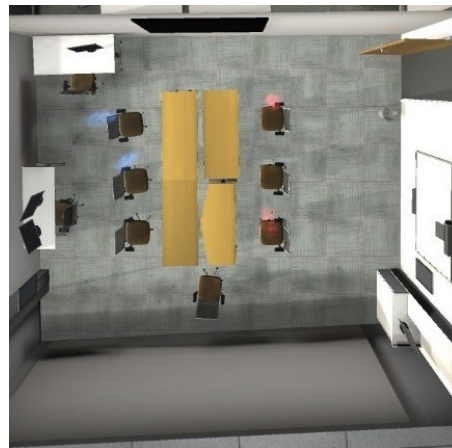
3.4 Creating shared space between remote locations

The motivation behind the project is to create a shared platform where people from different geographical location can come together. The idea of extending this project to set a platform for distance learning can be achieved by creating shared place between two remote buildings. As a prototype for this concept, about 28 network cameras are installed in Moss Art Center and Torgersen Hall class 1100, at Virginia Tech campus. The similar physical space is created at two different location to prototype this application, first is at observation room at Moss Art Center and second is Class 1100 at Torgersen Hall. The detected blob at physical space in Torgersen hall are projected to Observation Room, Moss Art Center in the virtual model. Fig. 3.8 shows two different avatars in the virtual model, where the blue ones are the avatars from people physically present in the observation room Fig.3.8(a) and red avatars are the people who are remotely present in the virtual model at Torgersen Hall Fig.3.8(b). This coordinate transformation is managed by the Blob manager on the server. The detail experiment about this application is covered in scenario three, evaluation section 4.4, chapter 4.



(a) Detection at Moss Art Center

(b) Detection at Torgson -Class 1100



(c) Shared Virtual environment

Figure 3.8: Remote presence applicaiton

3.5 Setup for collaborative object manipulation experiment

Installed cameras in the building can only be used to extract location information about the person present in the physical space. The experience of remote presence can be enhanced by extracting additional information like orientation, gesture performed by users. This information can be achieved by additional sensors like Kinect, Leap Motion controller, or motion tracking system (Cube, Moss Art Center). The collaborative object manipulation platform was created using additional sensor, Leap Motion controller. A study was conducted using to evaluate the effect of fidelity in collaborative object manipulation task. The setup for this study is shown in Fig. 3.3. The study detailed information and results for this study are covered in chapter 5

Chapter 4

Computer Vision Tracking

This chapter will mainly cover computer vision methodology used in Mirror Worlds project to extract the location information and also gives an overview about evaluation platform developed to compute detection accuracy. It will be structured as follows:

- Section 4.2, explains the proposed algorithm for human position detection based on the combination of detection results from background subtraction and convolutional neural network detector with some knowledge about the physical space (heuristics). Additionally, it describes the baseline algorithm developed in this project.
- Section 4.3, describes data association approach for human position tracking.
- Section 4.4, explains the evaluation strategy to compute the frame based detection accuracy measurement and sequential frame detection accuracy measurement [7] by comparing the results with ground truth data obtained from VATIC annotation tool.
- Section 4.5, describes the obtained results for three different scenarios. The sections also gives results for two baseline approaches and compares them to results obtained from proposed algorithm.



Figure 4.1: Flowchart of detection and tracking system

4.1 Introduction

The main objective of this task was to detect the moving people in the scene. The detection is very important step in recognition and tracking system. The flowchart shown in Fig. 4.1 elaborates the principle components of people detection and tracking.

4.2 Object Detection

4.2.1 Background subtraction for human position detection

Initially, the prototype of detection algorithm was developed in MATLAB [21] based on mixture of Gaussians [20]. This adaptive approach helps to eliminate the problem caused by static background models which is proposed to detect shadows by Prati [76]. The prototype was developed based on adaptive GMM implementation described in [77] [26]. The decision whether the pixel is part of background or part of foreground depends on the following ratio in equation 4.1,

$$\frac{p(BG|\vec{x}^{(t)})}{p(FG|\vec{x}^{(t)})} = \frac{p(\vec{x}^{(t)}|BG)p(BG)}{p(\vec{x}^{(t)}|FG)p(FG)} \quad (4.1)$$

where the BG and FG represents background and foreground respectively. If the above ratio in equation 4.1 is greater than 1 then the pixel is part of background and vice versa. As we cannot determine anything about the foreground object, $p(FG|\vec{x}^{(t)})$ is considered as uniform distribution and the decision is made by $p(BG|\vec{x}^{(t)})$.

$$p(\vec{x}^{(t)}|BG) > c_{thr} (= p(\vec{x}^{(t)}|FG)p(FG)/p(BG)) \quad (4.2)$$

In equation 4.2, c_{thr} is threshold value and $p(\vec{x}^{(t)}|BG)$ is background model. The background model is estimated on the training set of points represented by X . To adapt the changes in

light, weather conditions or changes in the scene, it is necessary to update the training set points by adding new samples and discarding old ones to estimate the dynamic background model. A suitable time T is selected for updates and hence at any point t will have $X_T = \{x^{(t)}, x^{(t-1)}, \dots, x^{(t-T)}\}$. The collected samples will have foreground objects as well as some background objects, hence the estimated of these models is denoted by $p(\vec{x}^{(t)}|X_T, BG+FG)$. These estimate is represented by GMM with M components given in equation 4.3,

$$p(\vec{x}^{(t)}|X_T, BG + FG) = \sum_{j=1}^M \omega_{j,t} * \eta(X_t|\mu_{j,t}, \sum_{j,t}) \quad (4.3)$$

where $\omega_{j,t}$: weight of j^{th} Gaussian in the mixture at time t , $\mu_{j,t}$ is the mean value of the j^{th} Gaussian in the mixture at time t , $\sum_{j,t}$ is the covariance matrix of j^{th} Gaussian in the mixture at time t , and η is Gaussian probability density function. In general M value is determined by the available computational memory and power, and commonly it is set to three or five. Similarly to expedite the computation the covariance matrix is set by equation 4.4 to consider the red, blue, and green to be independent and same variance in RGB image.

$$\sum_{j,t} = \sigma_j^2 I \quad (4.4)$$

The M distributions are ordered based on the fitness value ω_j/σ_j because the model with highest weight ω_j and lowest variance is most stable to represent the background. First K distributions are used as a background model. This value of K variable is determined by equation 4.5.

$$K = \arg \min_k \left(\sum_{j=1}^k w_j > T \right) \quad (4.5)$$

The threshold value T indicates the minimum prior probability the scene content background. Generally new pixel is determined whether it is part of background or foreground involves some important steps:

1. Check if new pixel x_t matches with any of the existing Gaussian models based on the distance
2. If match the pixel matches any of the i^{th} model then that model is adjusted to include this new data. The updates in the parameters are done by following equations:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \quad (4.6)$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (4.7)$$

where ρ is learning, rate given evaluated by equation 4.8.

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k) \quad (4.8)$$

In above equation 4.8, $1/\alpha \propto$ update rate of the model. All the other parameters for the unmatched models remains same.

3. If the new pixel is not matched with any of the models then it is considered as foreground pixel. The mixture model is adjusted by replacing the least probable distribution with new model with X_t as mean, initialized to high variance, and low priority. The prior weights of M distributions at time t are updated by equation 4.9.

$$\omega_{j,t} = (1 - \alpha)\omega_{j,t-1} + \alpha(M_{j,t}) \quad (4.9)$$

where $M_{j,t}$ is 1 for the matched models and 0 for the remaining ones. We implemented this algorithm and evaluated with package provided by MATLAB [21]. For our experiment, the M was set to 5 (default value) which signifies the support of different background modes.

The initial prototype was developed in MATLAB, which was difficult to scale for multiple buildings across campus. The scalable solution for this system was developed in OpenCV on linux platform. The algorithm was based on the background modeling algorithm to k-Nearest neighborhood classification algorithm [5]. The $k - NN$ model for background subtraction relies on the kernel density estimation. The density estimation starts by counting the k number of samples from fixed set of frames dataset X_T which lie within the volume V where T is amount of frames that need to considered in order to estimate the background. The volume V is hypersphere with diameter D . In our case, we decided T to keep as 50. The density estimation is given by

$$\hat{p}_{non-parametric}(\vec{x}|X_T, BG+FG) = \frac{1}{TV} \sum_{m=t-T}^t K\left(\frac{\|\vec{x}^{(m)} - \vec{x}\|}{D}\right) = \frac{k}{TV} \quad (4.10)$$

In equation 4.10, $K(u) = 1$, if $u < 1/2$ and 0 in other case. This kernel function can be selected based on the application. Some of the implementations have considered the smoother kernel functions where they have used Gaussian profile as kernel function [27]. Although the choice of kernel function is not as critical as compared to the choice of D [78]. In $k - NN$ based background subtraction the kernel estimation adapts the kernel size at each estimation of point \vec{x} . In the approach the width D of the kernel changes for each new estimate. The estimator is known as balloon estimator [5]. There are other estimators [79] available for this but balloon estimator is preferred most of the time because it is related $k - NN$ classification. In both, fixed kernel, and the balloon estimation of \vec{x} is made if there are more than k points in volume V . The only difference is selection of c_{thr} from 4.2.

In case of uniform kernel k is discrete and the estimates we get from it are discontinues whereas in case balloon estimator [5] k is fixed and volume V is the variable parameter. To differentiate the foreground objects from the background pixels indicators are added to equation 4.10,

$$\hat{p}_{non-parametric}(\vec{x}|X_T, BG) \approx \frac{1}{TV} \sum_{m=t-T}^t b^{(m)} K\left(\frac{\|\vec{x}^{(m)} - \vec{x}\|}{D}\right) = \frac{k}{TV} \quad (4.11)$$

where $b^{(m)}=1$ if it is classified as part of the background and 0 if it is classified as foreground object. These indicators are generated for all the points in the dataset X_T . Hence, the model estimated using this approach adapts the gradual changes in illumination, addition, and removal of some objects in the scene.

4.2.1.1 Pre-processing on the input frame

As mentioned in the section 3.1, configuration file has cropping coordinates listed for each camera. Whenever frame is received by the video capture object in OpenCV, it is cropped based on the cropping coordinates mentioned in the configuration file. This is done to reduce the overlapping regions between the 2 nearby cameras which will unnecessarily produce noise in the measurements. After the received frame is cropped, it is then passed to the background estimation model, BackgroundSubtractorKNN [77] to estimate the mask image.

The smoothing operation is a simple and frequently used for image preprocessing. The median filter with 3×3 size window runs through an image and replace each pixel with the median of its neighboring pixels. The square neighborhood around the pixel is considered to compute the median value at for single pixel. This square window size is known as kernel size. Further some morphological operations are performed to improve the quality of the obtained mask by performing dilate operation using the structuring element defined by 14×14 pixel size window.

4.2.1.2 Find contour form masked Image

After preprocessing the mask image obtain from backsubtractor, the contour is computed. The contour is curve joining all continuous points, having same color intensity. Detected contour can be visualized in Fig. 1.1. The contour detection algorithm [80] is applied on the mask image obtained from background subtractor. These detected contours are stored as vector of points. Each contour k , from the detected contours, has many elements such as `hierarchy[k][0]`, `hierarchy[k][1]`, `hierarchy[k][2]`, and `hierarchy[k][3]`. All these are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour respectively. If there are no other contours found near k^{th} contour then `hierarchy[k]` will be negative. In our system, we are using retrieval algorithm which only retrieves extreme contours, it is denoted by *RETR_EXTERNAL* enumeration in OpenCV API. It sets `hierarchy[k][2] = hierarchy[k][3] = -1` for all the contours. For representing these detected contours, we are using an algorithm that compresses horizontal, vertical, and diagonal segments and only extracts the end points. This is represented as *CHAIN_APPROX_SIMPLE* enumeration in find contour [80] algorithm in OpenCV. This detected contour vector is then passed to a filter function in our algorithm where the contour with $area \geq contourMinThresh$ and $area < contourMaxThresh$ are considered as valid detection and other contours are erased from the vector.

4.2.1.3 Combining blobs with Nearest neighbor

In addition to background subtraction, an algorithm was added to aggregate the small blob together to form a big blob [81] [82]. The common algorithm to for finding nearest-neighbors is kd-tree [83] which works well with low dimensional data but its performance degrades with higher dimensional data. The Flann [82] algorithm is based on randomized kd-tree and hierarchical k-means tree algorithm. Randomized kd-tree maintained a priority queue across all randomized tree to order the search by increasing distance to each bin boundary. While, hierarchical k-means tree does single traversal through the tree and adds priority queue for all unexplored branches. The Flann algorithm is used to combine all the small contours to a bigger contour in proximity. The algorithm can be summarized in Fig. 4.2, After detecting all the blobs, kd-tree of all the centroids for these contours locations is generated. A copy of all the centroid locations is created in another vector and it is sorted based on the area detected for that contour. Then query contour is the contour with largest radius. The number of neighbors searched are $< nnMaxContours$. The algorithm will loop through nearest neighbors results to combine them based on the distance of contour from the query point. The decision can be made by equation 4.12,

$$distance \leq maxDistThresh \quad (4.12)$$

where, $maxDistThresh$ can be computed by equation 4.13,

$$maxDistThresh = queryRadii + currNbrRadii + searchRadii \quad (4.13)$$

where, $currNbrRadii$ is the radius of the processing contour radius from the nearest neighbor search result, and $searchRadii$ is maximum distance set for a search space around the query contour location. Once all valid contours which are ready for combining with the query point are collected, convex hull algorithm [84] is applied to combine these small contours.

4.2.2 Detection with neural networks

The conventional background subtraction algorithm mentioned in the above section traditionally works on background modelling with color, intensity, or gradient s for each pixel

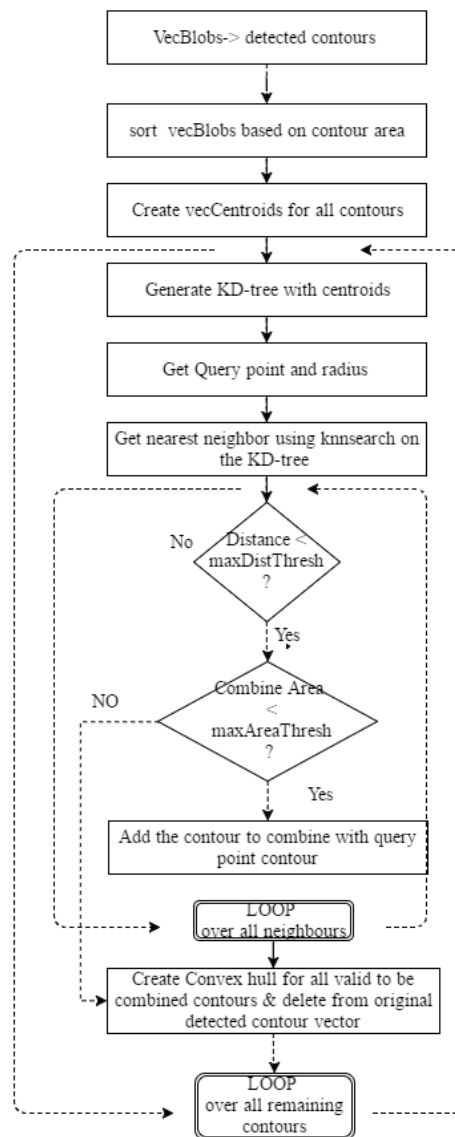


Figure 4.2: Combine small contours flowchart

using k-nearest neighbor method [77], mixture of Gaussians [20] to become slightly invariant to illumination changes. As the algorithm depends on the environmental aspects the accuracy with independent conventional background subtraction approach is not satisfactory. Hence, the state-of-the-art deep learning perception algorithms and specially convolutional neural network (CNN) based detectors [85] [86] were explored, which has improved the object detection task significantly over last decade [6] [22] [87] [88].

4.2.2.1 Convolutional neural networks

The training is done for detecting the predefined object types and for each object type large data set is required. The objective of our system was to detect the human in the frame obtained from the input camera. The human perception obtained through the camera is not static its dynamic with the changing background. Hence, the issues like occlusions, faster movements, dynamic posture makes it hard problem for detections. The YOLO (You Only Look Once) framework [6] [17] based on convolutional neural network was used in proposed system as person detector.

Most of the current classification systems use classifiers to perform detections. The common workflow for detection is to detect object, then use the classifier for that object and evaluate it various locations in the image and scale in it test image. In comparison with deformable parts models (DPM) [34], where a sliding window approach is with classifier run over evenly spaced locations in entire image, YOLO detects objects with much faster. Similarly, it outperforms the R-CNN based detectors [22] in speed for detections. R-CNN based detectors first create bounding boxes for potential detections and then classifier is executed on these proposed boxes. After this classification is over, the bounding boxes are refined by removing the duplicate detections and rescore the bounding boxes based on the other objects in the scene. In DPM [34] and R-CNN [22], complex pipeline is involved in classifying the objects and hence, it is difficult to optimize because each individual component must be trained separately. The 3 important aspects because we selected YOLO over other approaches, first, it is extremely fast. Second, it sees the entire image during train and test time, which helps to get a better perspective about the object and full contextual information about the classes

and their appearance. Third, it is generalizable. In our system, we needed something real-time, and can adapt the various posture of human body with dynamic background. Hence, explored this state-of-the-art detection algorithm, YOLO [6].

4.2.2.2 Network Design

YOLO [6] predicts all the bounding boxes for all the classes simultaneously because it uses features from entire image for training as well as for testing. This design enables complete training and real-time speeds while better average precision in detection. In this algorithm, image is divided into $N \times N$ grid and B bounding boxes with confidence scores are computed for each cell in this grid. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the box is that it predicts. At test time, class specific confidence is computed by equation 4.14, multiplying conditional class probabilities and individual confidence predictions.

$$\Pr(Class_k|Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_k) * IOU_{pred}^{truth} \quad (4.14)$$

where, IOU_{pred}^{truth} is intersection between predicted bounding box and ground truth, $\Pr(Class_k|Object)$ represents the conditional class probabilities for each cell (C for each cell).

The network architecture for YOLO is inspired from GoogLeNet model for image classification [42]. In our system, we are using 24 layer convolutional neural layers followed by 2 fully connected layers. These layers are trained on ImageNet 1000-class competition dataset [89]. This model is then converted for detection task, by adding 4 convolutional layer and two fully connected layers with randomly initialized weights to this network to enhance the detection performance [90]. The network uses 448×448 resolution for detection. The final layer in the network is used for predicting the class probabilities and bounding box location information. All other layers in the network uses leaky rectified linear activation function given by equation 4.15,

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \quad (4.15)$$

In this training sum squared error is used for optimization purpose to keep it simple. If cell does not contain any object then the confidence score is set to zero for that cell, this generates a large contrast for cell containing objects which leads to instability of the model. To counter this issue, the more weightage is given to loss due to bounding box coordinate predictions and less weightage to confidence predictions for bounding boxes that does not contain any objects. The use of λ_{coord} and λ_{noobj} to achieve this improvement. During the training of this network, the loss function is used is given by equation 4.16,

$$\begin{aligned} & \lambda_{coord} \sum_{i=0}^{N^2} \sum_{j=0}^B I_{i,j}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{coord} \sum_{i=0}^{N^2} \sum_{j=0}^B I_{i,j}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\ & + \sum_{i=0}^{N^2} \sum_{j=0}^B I_{i,j}^{obj} (C_i - \hat{C}_i)^2 \\ & + \lambda_{noobj} \sum_{i=0}^{N^2} \sum_{j=0}^B I_{i,j}^{obj} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{N^2} I_{i,j}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \quad (4.16)$$

where, I_i^{obj} indicates if there is any object in the cell i and $I_{i,j}^{obj}$ indicates the j^{th} bounding box predictor in the cell i is responsible for prediction. This loss function penalizes the classification error from the object in the grid cell and penalizes bounding box coordinates if predictor is responsible for ground truth detection.

4.2.3 Combine approach based on background subtraction and Neural Network

The procedure to combine the background subtraction and convolutional neural network method is depicted in Fig. 4.3,

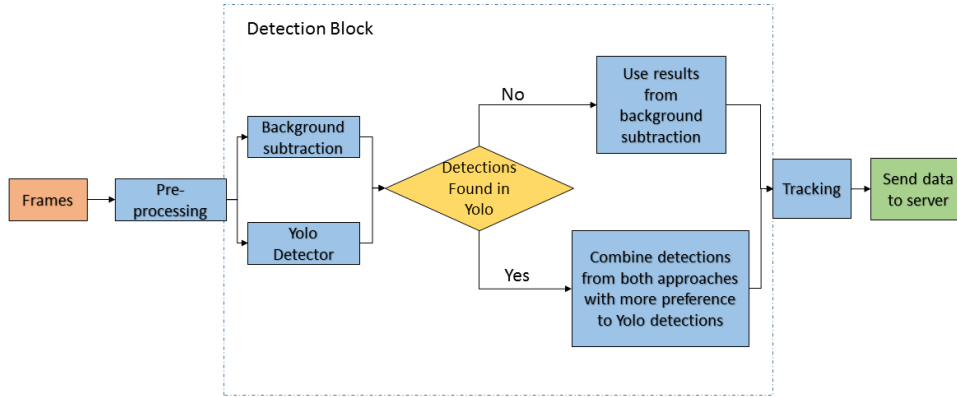


Figure 4.3: Combine workflow for detection in MirrorWorld

As shown in Fig. 4.3, input frames are pre-processed using morphological operations. This processed output was given as input to detect block. The proposed method considers detection results obtained from both the approaches. If there detections obtained using neural networks then results from background subtraction were combined with YOLO detector output. While combining these results obtained from both approaches, more preference was given to result obtained YOLO detector. For example, in Fig.4.4, consider that blob id one is detected using background subtraction and YOLO detector both approaches while blob id two is detected only using background subtraction. In this scenario, more preference was given to bounding box results obtained from convolutional neural network detection approach for blob id one and the detection given by background subtraction method for blob id one was discarded. All doubly detected blobs were filtered based on the overlap threshold criteria and then remaining results from background subtraction were combined with the filtered output. On the other hand, if there are zero detection found in YOLO detector then algorithm only considers the background subtraction results for tracking.

4.3 Human Position tracking

In the previous section, different human position detection methods like conventional background subtraction, YOLO detector [17] and a combined approach of YOLO detector and

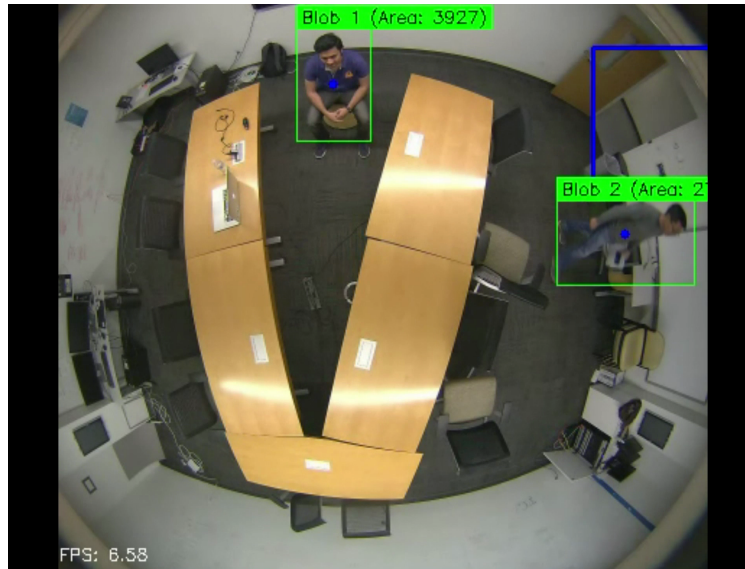


Figure 4.4:

background subtraction are discussed. The detection only involves verifying the presence of human in any single frame, while tracking involves monitoring the temporal and spatial changes in throughout the video sequence, which includes presence, position, and size related information. This is done by matching the target region through the sequence of images in video [91].

4.3.1 Data Association

The process of matching current observed objects information with previously observed objects. These algorithm tries to find matches that optimize certain match criteria. Hence it can be formulated as solving a constrained optimization problem. To match the current measurement, we should keep record of the previous detected locations of the object. The element which keeps track of temporal history of certain object is known as track. This track element can be used to get the information about objects last known location and its current velocity. In our system, this track object is designed to keep track of *track_id*, *age*, *visibility_count*, *invisible_age*, *visible_flag*, *bounding_box*, *contour*, and *prediction* information. The track object can be represented as following structure shown in Fig.4.5,

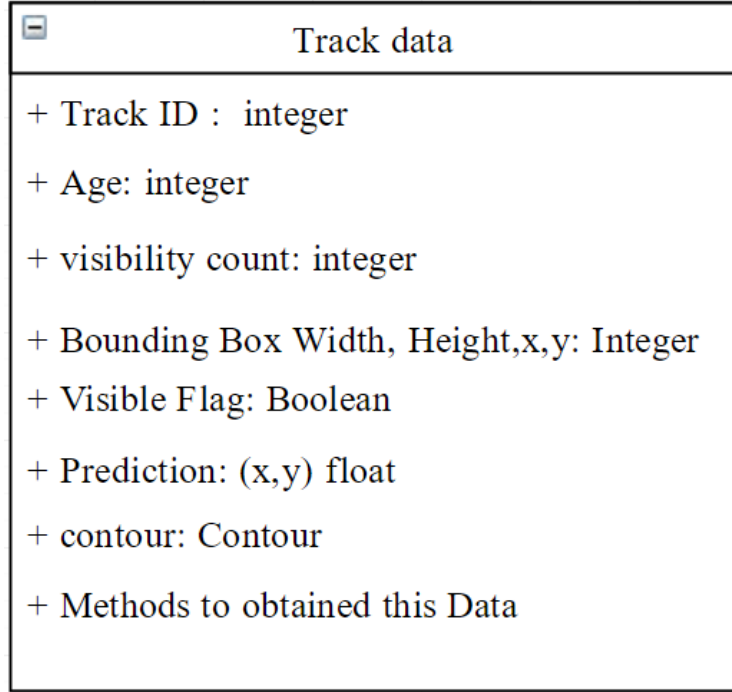


Figure 4.5: Track object structure

The next step is to match this detected current blob measurement spatial information to the previously generated tracks and the prediction value in the track object. The assignment decision is done through Track assignment workflow given in Fig. 4.6. This is decided based on the cost to assign the blob to certain task and the cost is determined by degree of overlap function. Some unrealistic detection is ignored by the upper threshold on the cost to assign blobs to track. The cost is computed by equation 4.17.

$$cost = \sqrt{(blob.x - prediciton.x)^2 + (blob.y - prediciton.y)^2} \quad (4.17)$$

4.3.2 Track Management

The track management is an important aspect of this algorithm, which was based on the few attributes of the track object. The track management process can be represented by flowchart given in Fig. 4.7 The value of *visibility_count* is incremented in every frame if the detection algorithm finds human blob in the frame and it is assigned to one of the existing

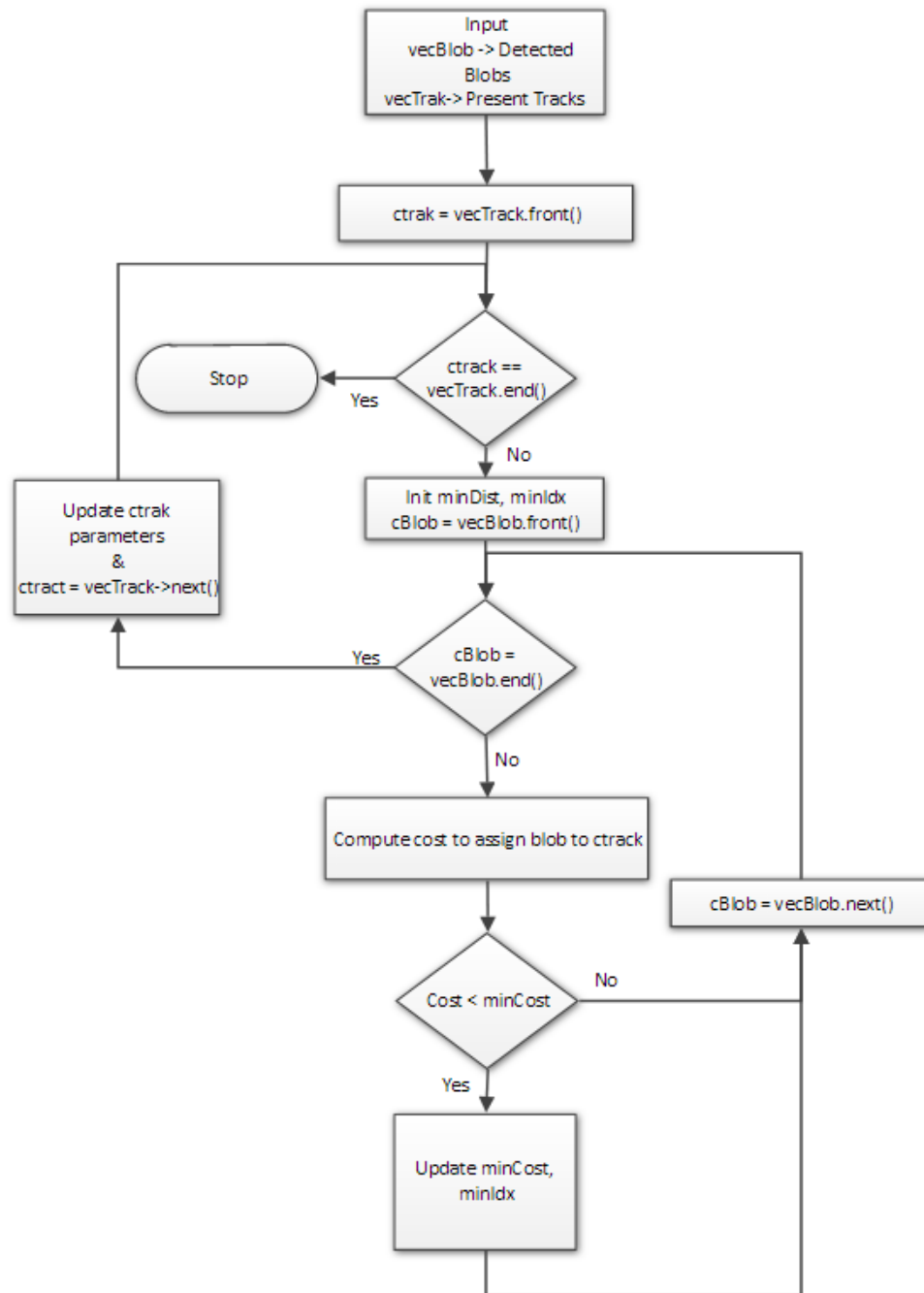


Figure 4.6: Track assignment flowchart

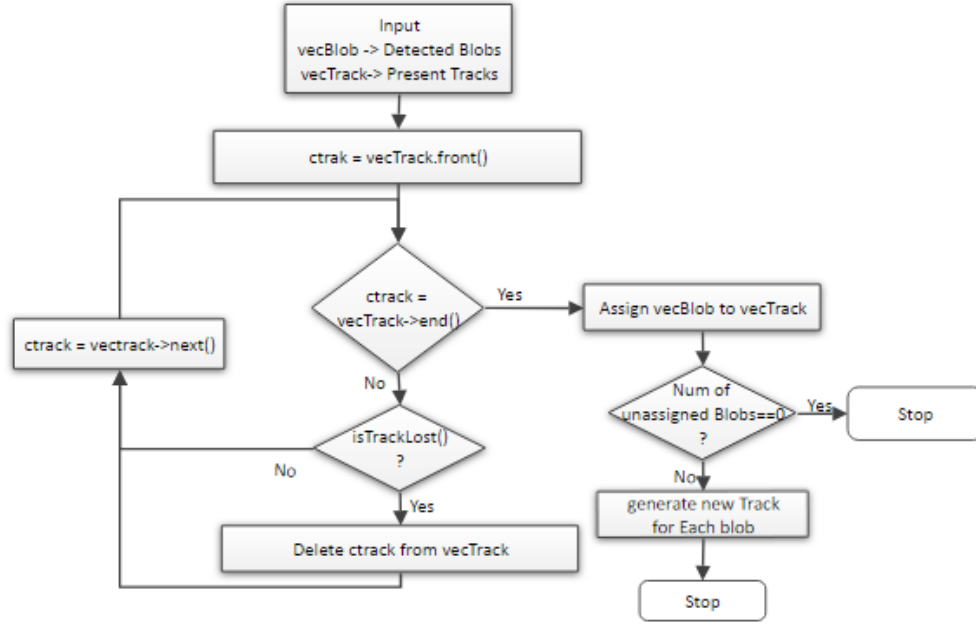


Figure 4.7: Track management flowchart

track. The blob is assigned to certain track if and only if, the cost computed for to assign certain blob to a track satisfies condition given by equation 4.18 and the workflow for track assignment process is given in Fig. 4.6

$$cost > noAssignmentCost \quad (4.18)$$

There are three different scenarios for updating the track parameters,

- Number of detected blobs = Number of tracks present and *both* $\neq 0$

In this case, as *detectedBlobsSize* > 0 and algorithm will try to assign these blobs to existing tracks in the current state. If a certain track is assigned to a detected blob after satisfying (4.18), then the track object parameters are updated with resetting the *invisible_age* value to 0 and incrementing the *visibility_count*.

- Number of detected blobs $>$ Number of tracks present and *both* $\neq 0$

In this case also, *detectedBlobsSize* > 0 and algorithm will still try to assign these

blobs to existing tracks in the current state. For all the tracks, satisfying (4.18), are updated with resetting the *invisible_age* value to 0 and incrementing the *visibility_count*. But as there are more number of blobs detected in the frame, some blobs will remain unassigned after the track assignment operation completed. New tracks will be generated for all such blobs with new *Contour* objects.

- Number of detected blobs < Number of tracks present and *both* $\neq 0$

From the vector of detected blobs, algorithm tries to assign these blobs to existing tracks in the current state. For all the tracks, satisfying (4.18), are updated with resetting the *invisible_age* value to 0 and incrementing the *visibility_count*. But as there are more number of track present, hence some tracks will remain unassigned to any of the detections. In this, track parameters are updated by incrementing *invisible_age* and *age* values.

These updates are very important track management. Each time before track assignment operation is called on the detected contours, the tracks are deleted based on the *visible_percent* and *invisible_age*. The conditions can be represented by equation 4.19.

$$\begin{aligned}
 \text{visible_precent} &= \frac{\text{visibility_count}}{\text{age}} \\
 \text{visible_precent} &> \text{visible_threshold} \\
 &\text{or} \\
 \text{invisible_age} &> \text{invisible_max}
 \end{aligned} \tag{4.19}$$

When the conditions in equation 4.19 are satisfied for certain track, then that track is marked as “LOST” and it will be deleted from the track vector.

4.3.3 Prediction

Initially in this system, the prediction value was estimated same as the previous location value, that means that detected blob will be stationary or it will have very less movements between the frames. However, smoother tracking for multiple objects can be achieved by

estimating blob future location based on the velocity and current measurement parameters. The Kalman Filter [92] algorithm was used for state prediction in this system. The algorithm tries to solve the state estimation problem of a discrete-time controlled process which is governed by the differential equations given by 4.20,4.21

$$\overline{X}^t = A^t X^{t-1} + B^t U^t + \varepsilon^t \quad (4.20)$$

$$\overline{Z}^t = C^t X^{t-1} + \delta_t \quad (4.21)$$

where \overline{X}^t and X^{t-1} is current and previous state vectors, U^t is control input, and current measurement is denoted by Z_t . ε^t and δ^t are random vectors representing the uncertainty in the measurement. Given the previous state, $X^{t-1} \propto N(\mu_t, \Sigma_{t-1})$, current state X^t can be estimated through prediction and correction term mentioned in 4.22,

$$\begin{aligned} \overline{\mu}^t &= A^t \overline{\mu}^{t-1} + B^t U^t \\ \overline{\Sigma}^t &= A^t \overline{\Sigma}^{t-1} (A^t)^T + R^t \\ K^t &= \overline{\Sigma}^t (C^t)^T (C^t \overline{\Sigma}^t (C^t)^T + Q^t)^{-1} \\ \mu^t &= \overline{\mu}^t + K(Z^t - C^t \overline{\mu}^t) \\ \Sigma^t &= (I - K^t C^t) \overline{\Sigma}^t \end{aligned} \quad (4.22)$$

In our system, Z^t , measurement is done by detecting the contour location and calculating the centroid of the contour using moments feature. These moments can be obtained by 4.23,

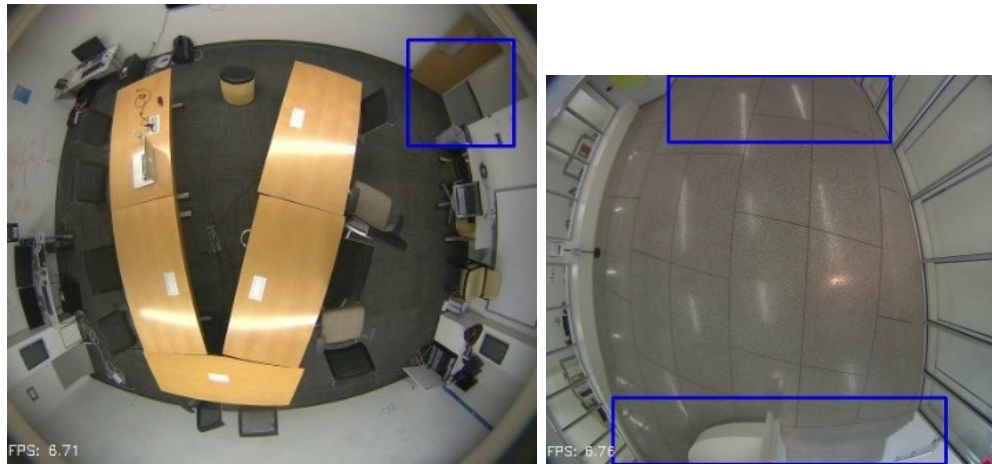
$$M_{i,j} = \sum_x \sum_y x^i y^j I(x, y) \quad (4.23)$$

where, $i, j = 0, 1, 2, 3..$ and $I(x, y)$ are pixel intensities and centroid $\{\overline{x}, \overline{y}\}$ is computed by 4.24,

$$\{\overline{x}, \overline{y}\} = \{M_{10}/M_{00}, M_{01}/M_{00}\} \quad (4.24)$$

4.3.4 Combined heuristics based tracking algorithm

The proposed track management algorithm was modified some heuristics about the physical space knowledge. The system uses detection results obtained from combine approach discussed in section 4.2.3. There are 3 important heuristics that were added to the system to improve the performance.



(a) Spawn area in observation room (MAC)

(b) Spawn areas in hallway

Figure 4.8: Heuristics based on Spawn area, region marked with blue rectangles

- Include a spawn area for each camera to generate the new tracks
- Delete the track if there are no detection in coming frames over certain threshold
- Boost the neural network detection class for humans based the track of number of blobs present in the certain region

4.3.4.1 Spawning Area heuristics

The tracking system discussed in section 4.3.2 generates new tracks if there are unmatched contours after the track assignment task. Due the constraint environment, in this system, the possible entry and exit locations for the participants was fixed. This indicates that a new track ideally cannot be generated anywhere else than the entry and exit locations. The tracking algorithm was modified with these constraints. New tracks were only generated if the detected location of blob lies within the spawn area.

The spawn region was given as input to system as rectangular coordinates in configuration file for each camera. The spawn region is shown in Fig. 4.8. The change in track management process can be summarized in Fig. 4.9,

Now the new track can only be generated in if the detected blob origin lies within the

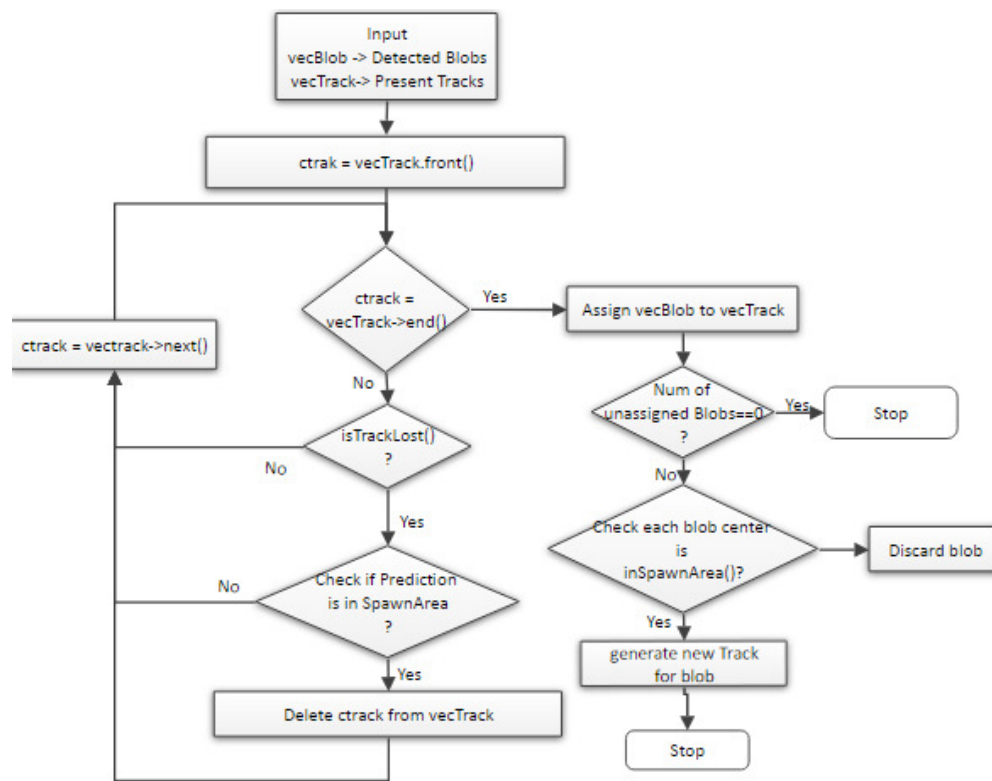


Figure 4.9: Heuristics based track management

rectangular spawning region. Similarly, the track can be deleted if track prediction for the track object lies in rectangular spawning region and it is masked as “LOST” by satisfying the conditions in 4.19.

4.3.4.2 Delete track on No detections found

The person entered in the camera region can only exit through the known exit points. Hence, tracks were deleted only if the track prediction lies in spawn area and it is marked “LOST”. The “LOST” condition is true if the parameters of track objects satisfy 4.19.

4.3.4.3 Boosting Neural Network Detection

In previous section 4.2.3, the combined approach for human detections based on background subtraction and YOLO [17] detector was used for detecting foreground objects. The spawning area heuristics is used to set and reset the *blobExistFlag* to indicate the presence of the human in certain camera region. This flag is passed as input to the detector function. This detector function returns probable object class which has maximum confidence score within frame. With heuristics, if there is presence detected, then confidence score for person class is increased by certain fraction to increase probable location given by YOLO detector [17].

4.4 Evaluation

Mirror Worlds infrastructure supports future studies related to computer vision, Virtual reality, remote collaboration, learning science, etc. Hence, the system should be reliable, robust, and real-time. The implemetation of algorithm was done in C++ on Intel Xeon(R) CPU E5-2687W v3 @ 3.10GHz 20 processor with Quadro M4000/PCIe/SSE2 graphic card. The video of 1M (1056× 960) is given as an input to this system at 15fps Since no annotated people video dataset was publicly available with overhead view from omni-directional cameras, the proposed system was evaluated with three recorded video sequences. These videos were annotated manually to test the performance of the approach. As, Mirror Worlds

project provides platform for variety of applications and possible cases, the measurement of the performance was done with 3 main scenarios to cover these areas.

- Constraint environment: single camera and two people in Observation Room (MAC)
- Indoor but less constraint environments: single camera, people moving in corridor
- Remote presence scenario: two cameras at different geographic location, people moving in shared Virtual place

The performance evaluation metrics was developed to perform the quantitative analysis of detection algorithms. The system is mainly evaluated on two metrics. First is Frame Detection Accuracy (FDA) and second is Sequential Frame Detection Accuracy (SFDA). The evaluation was done by comparing the detection results with ground truth bounding boxes data obtained for each frame.

4.4.1 Extracting the Ground Truth Data

The ground truth data was obtained by using VATIC (Video Annotation Tool from Irvine, California) [93] [94] tool. Most of the tools include computer vision and machine learning methods that support human annotations [95] [38] [96] [97]. VATIC was developed based on the idea of Sorokin and Forsyth [98], where they proposed the image labeling task can be crowdsourced at low cost from Amazon Mechanical Turk (MTurk). It provides the open platform monetarized crowdsource video labeling where interpolation is done by nonlinear least-cost paths with efficient dynamic programming algorithms based on image data and user annotated endpoints. The platform really fits the requirement and future prospects of enabling the other researcher outside to explore and use the Mirror World infrastructure to conduct the research in virtual reality and computer vision domain. VATIC uses MTurk platform to find the people who can annotate the video. The user interface (UI) for the tool is interactive browser video player that helps user to label objects in throughout the video sequence. The interface is developed using javascript language. The work-flow for performing video annotation task can be summarized as Fig. 4.10, In the above work-flow, all the frames

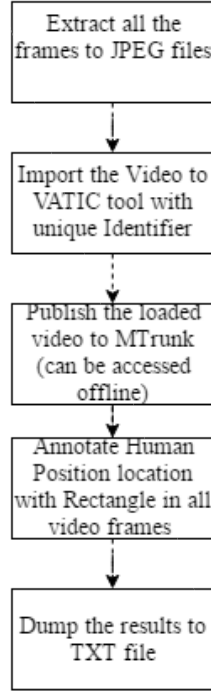


Figure 4.10: VATIC video annotation flowchart for our system

are extracted in separate jpeg encoded files. Then this extracted file system should be loaded to VATIC tool with unique identifier. As the interface for the tool is based on flash codec and implemented in Javascript language, enables it to perform smoother and quicker annotation process. The annotation process is having ability to interpolate between the sparse set of annotations. The user is asked to annotate every T frames and tool interpolates the path for marked object between these frames. The bounding box coordinates are represented as vector,

$$b_k^t = [x_1, x_2, y_1, y_2] \quad (4.25)$$

where, b_k^t is k^{th} bounding box coordinate at time t . If user has marked every T frame in the video then tool interpolates the b_k for all frames between t and $t + T$ by linear interpolation method given by equation 4.26.

$$b_k^{t,lin} = \left(\frac{t}{T}\right)b_k^0 + \left(\frac{T-t}{T}\right)b_k^T \quad (4.26)$$

It is also important to extract the visual model of the tracked object to interpolate the object accurately. The system uses discriminative classifier trained to produce high scores

VATIC annotation results
+ Track ID: Track Number
+ xmin: top left x-coordinate
+ ymin: top left y-coordinate
+ xmax: bottom right x-coordinate
+ ymax: bottom right y-coordinate
+ frame: frame Number
+ lost: flag to indicate annotation outside window
+ Occluded: flag to indicate annotation is occluded
+ generated: flag to indicate annotation is interpolated
+ label: class name

Figure 4.11: VATIC annotation result structure

on positive bounding boxes and low scores on the negatives. For each of the marked blob by the user, feature descriptor is computed for each b_k object which is combination of HOF and color histogram features represented as,

$$\phi_k(b_k) = \begin{bmatrix} HOG \\ RGB \end{bmatrix} \quad (4.27)$$

with these features and labelled data $y_k \in \{-1, 1\}$, a linear SVM is classifier is trained to predict the location of the unannotated frames. Once the annotation process for entire video frames is complete, all bounding box information is dumped to text file, which can be used as an input to in evaluation platform. The process is explained in section 4.4.2. The extracted information has following structure shown in Fig.4.11.

From this result, lost, occluded, and generated flags are used to determine the valid detections or presence of the human in the rectangular window, frame number is used to compare the results, and bounding box coordinates to find match or spatial accuracy of our detections.

4.4.2 Detection Measurement

The goal of developing the performance measure tool was to give importance to both, the overall detection accuracy, and goodness of the detection. The accuracy of the detection is indicated by number of objects detected, missed detects, false positives, and false negatives, whereas the goodness of the accuracy is measured by calculating the spatial overlap of the detection with the ground truth data. Hence, an evaluating platform was developed, which will give us Frame Detection Accuracy (FDA) and Sequence Frame Detection Accuracy (SFDA) [7]. The SFDA value signifies a frame level measure to evaluate spatial alignment match of the detections with the ground truth data, along with missed objects, false alarms count. The Frame Detection Accuracy (FDA) is measured for each frame in the video by calculating the spatial overlap between the ground truth and output of detected human position as a ratio of the spatial intersection between the two objects and the spatial union of them. All overlaps are summed and normalized over the average of total detected objects and ground truth objects.

$$FDA_t = \frac{\text{Overlap Ratio}}{\left[\frac{N_G^t + N_D^t}{2}\right]} \quad (4.28)$$

where, N_G^t is number of ground truth objects, N_D^t is number of detected objects in frame t , and overlap ratio is defined in equation 4.29,

$$\text{Overlap ratio} = \sum_{i=1}^{N_{mapped}^t} \frac{|G_i^t \cap D_i^t|}{|G_i^t \cup D_i^t|} \quad (4.29)$$

where, G_i^t denotes the i^{th} ground object in t^{th} frame, D_i^t denotes the i^{th} ground object in t^{th} frame, N_{mapped}^t is number of mapped ground truth objects and detected objects in frame t . In order to measure performance for the whole video sequence, FDA value is calculated for all the frames. For normalizing only the frames with which has the objects either in ground truth or in detected objects. This measure considers the false alarms as well as missed objects. Hence, Sequence Frame Detection Accuracy (SFDA) can be denoted by (4.30),

$$SFDA = \frac{\sum_{t=1}^{t=N_{frames}} FDA_t}{\sum_{t=1}^{t=N_{frames}} \exists(N_G^t \text{ OR } N_D^t)} \quad (4.30)$$

where, N_{frames} is total number of frames in the video sequence. The ground truth annotation process is done manually. Hence, the correctness of bounding box value will depend on the accuracy in detections. To compensate annotation error, thresholding approach for spatial accuracy measurement [7] was adapted in evaluation method. The detected object can be classified as correctly detected even if it is partially overlapped with ground truth object. For this system, Ovr_Thresh is kept as 50% in equation 4.32. The computation of this partial spatial accuracy can be given by equation 4.31,

$$Overlap\ Ratio\ Thresholded = \sum_{k=1}^{N_{mapped}^t} \frac{Ovr_Thresh(G_i^t, D_i^t)}{|G_i^t \cup D_i^t|} \quad (4.31)$$

and,

$$Ovr_Thresh(G_i^t, D_i^t) = \begin{cases} |G_i^t \cup D_i^t|, & \text{if } \frac{|G_i^t \cap D_i^t|}{|G_i^t|} \geq Ovr_thresh \\ |G_i^t \cap D_i^t|, & \text{Otherwise} \end{cases} \quad (4.32)$$

4.5 Results

This sections explains, the strategy for evaluating the proposed algorithm. Three different scenarios were collected to evaluate the proposed algorithm. Each scenario video was approximately 3minute in length containing total of average 4000 frames. These scenarios were evaluated with conventional background subtraction algorithm, background subtraction with heuristics and combined algorithm based on background subtraction and Yolo detector. The evaluation platform computes FDA , and $SFDA$ [7] value for all frames and video sequences respectively. Precision and recall values were also calculated based on the values of false positives, false negatives, and true positives. These values can be represented can be represented as confusion matrix, Table 4.1. Precision value was computed by equation 4.33,

$$Precision = TP / (TP + FP) \quad (4.33)$$

where, TP is true positives, FP is false positives.

Table 4.1: Confusion Matrix

	Ground truth Blob exists at (x, y)	Ground truth Blob does not exist at (x,y)
Algorithm A detects as Blob at (x, y)	TP	FP
Algorithm A does not detect a Blob	FN	TN

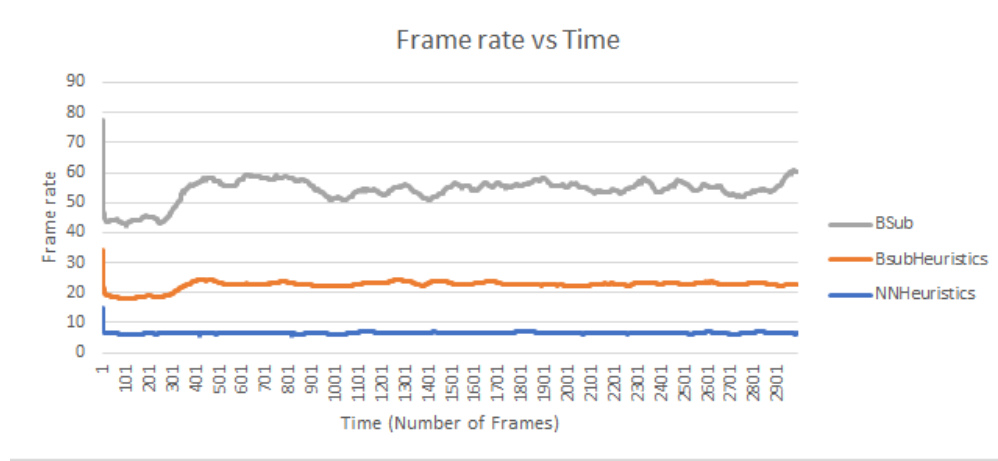


Figure 4.12: Comparison of 3 approaches - Frame Rate vs Time

The obtained values for TP, FP, FN were used to calculate False positive rate (FPR), False Negative Rate (FNR) for this proposed algorithm. The rate can be computed by equation 4.34,

$$\begin{aligned} FPR &= FP / (FP + TN) \\ FNR &= FN / (TP + FN) \end{aligned} \quad (4.34)$$

Along with accuracy measure it was important to see how this algorithm performs over other baseline approaches. The frames per second (*fps*) values were recorded for 2975 for baseline line approaches background subtraction (BSub), background subtraction with heuristics (BsubHeuristics) and proposed combine algorithm (NNHeuristics). The analysis result for for video sequence of 2975 frames is shown in Fig.4.12.

Selected three scenarios includes four different video sequences. Table 4.2, shows some from

collected video sequences. These frames were selected such that it will give an approximate idea about the performance of algorithms over entire video sequence.

In this Table. 4.2, column a, is for scenario I. In this scenario, the video was collected with 2 people moving in the constraint environment. One participant was sitting at the center, while other participant was moving around the room. This scenario mainly illustrates the challenges faced in background subtraction based algorithms and how it can be eliminated by proposed solution. Column b, contains frames extracted from video sequence II. This video sequence was recorded for less constraints environment (corridor in building) where people motion was not restricted and environment was dynamic with occlusions between different participants. This scenario mainly covers the challenges involved in human position detection in overhead fish eye lens. The overhead view obtained from the fish eye lens makes difficult to detect the people near the corners and edges of the image. Column c, has frames extracted from 2 different video sequences. Both video sequences are again in constraint physical spaces but at different location. This scenario demonstrates the application of proposed algorithm for remote presence. The detected human position at two different physical location was mapped to a shared virtual environment.

These recorded video sequences were annotated using VATIC tool to obtain the ground truth data. This ground truth labeled data was used to evaluate the performance of algorithms over entire video sequence.

4.5.1 Scenario I

Experimental results for all three approaches on video sequence in constraint environment is given in Table.4.3.

Background subtraction method (Table.4.2.1, Approach (A)), fails in detecting the target if the target is stationary for longer time. The background model adapts to foreground object's pixel intensity values over time and it becomes part of the the background model (Table.4.6, column (a), 4th frame). The first frame in each column corresponds to frame after background learning process is completed. The false positives in this case can be avoided

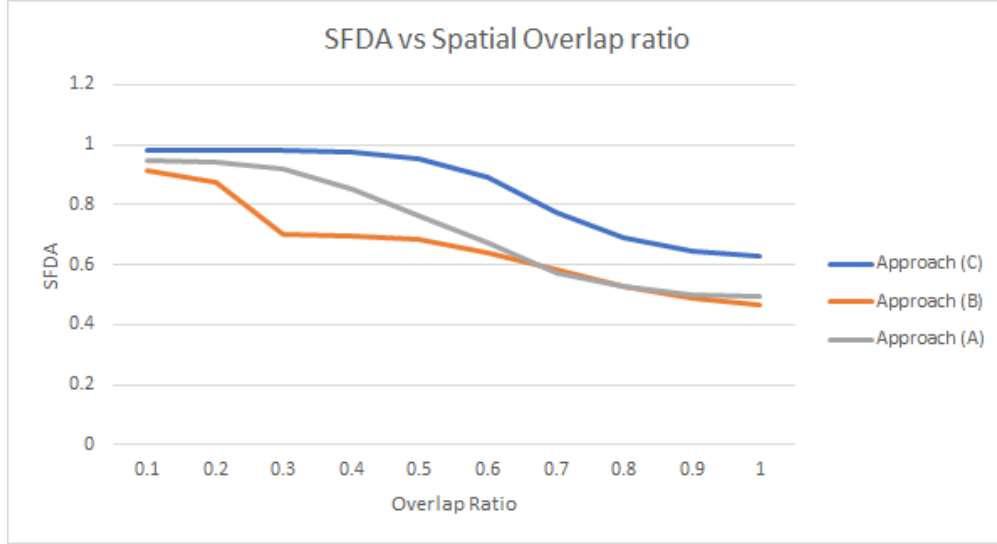


Figure 4.13: SFDA vs Spatial Overlap Ratio threshold ($Ovlap_thresh$)

with addition of heuristics. The heuristics also helps to eliminate other false blobs which might arise due to environmental changes. In column b, 4th frame, detected blob has very less spatial overlap with the actual human position. In this case, target is not detected by background subtraction but added heuristics in this approach keeps track of target previous location. The 3rd column has experimental results for combined approach of background subtraction and YOLO detector with heuristics for target detection. The results listed this column (c) indicates that the proposed solution was able to detect stationary target with maximum possible overlap with target position and area.

The FDA and $SFDA$ values were computed for entire video sequence. The results shown in Table.4.4 were obtained by keeping intersection over union (IOU) threshold value in equation 4.32 to 50%.

The combined approach based on neural network and background subtraction with heuristics performed better in term of spatial accuracy value across complete video sequence. The value of $SFDA$ is computed by equation 4.30.

The effect of variation of $Ovlap_thresh$ in equation 4.32 on $SFDA$ value can be illustrated with Fig.4.13.

4.5.2 Scenario II

Experimental results for all three approaches on recorded video sequence in less constraint environment is given in Table.4.6. In this scenario, three users are simply walking through the corridor. The recorded video has target entering into the camera frame with occlusion. Overall this scenario covers challenges involved in detection and tracking of fast walking people in dynamic environment.

In this scenario, baseline approach (A) performed slightly better than the proposed solution and baseline approach (B). In this case, multiple targets entered into the camera frame simultaneously with one occluding the other. Hence, heuristics based approaches sometimes failed to detect all present participants. Overall, the performance of all three approaches was comparable, with approach (A) performing slightly better than others.

4.5.3 Scenario III

This case demonstrates the application of our system. Table.4.7, two images in first row, have detection results obtained from evaluating the combined algorithm based on background subtraction and neural network, the results were evaluated over two video sequences recorded at two different location in the campus (Moss Art Center (Room I) and Torgersen Hall (Room II)). Second row in image, displays 3D shared virtual environment. In this task, the detected human position was mapped to an avatar (blue and red blobs) in the virtual model. The blue avatar is a representation of human position detected in room I at Moss Art Center, while red avatar is representation of detected position in room II at Torgersen Hall. These two video sequences were separately evaluated for frame based detection accuracy measurement. Table.4.8, shows the measurement values obtained for video sequence in Table.4.7, column (a), while Table.4.9, shows measurements for video sequence in Table.4.7, column (b).

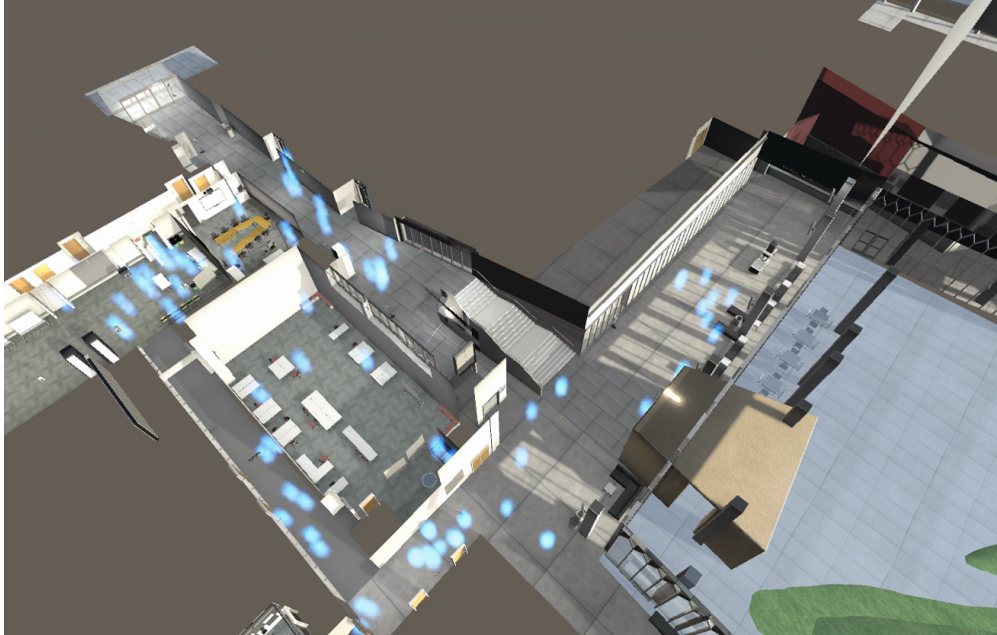


Figure 4.14: Overview of the entire Moss Art Center 3D model with avatars displaying real-time position of people present in the building

4.5.4 Scaled system

The developed algorithm was extended to entire building in Moss Art Center and Torgersen Hall to support all the installed cameras in the buildings. The Fig.4.14 shows scaled version of the system with 28 working cameras in Moss Art Center tracking the real-time population and mapping these positional information in the 3D environment.

Table 4.2: Frames for different scenarios

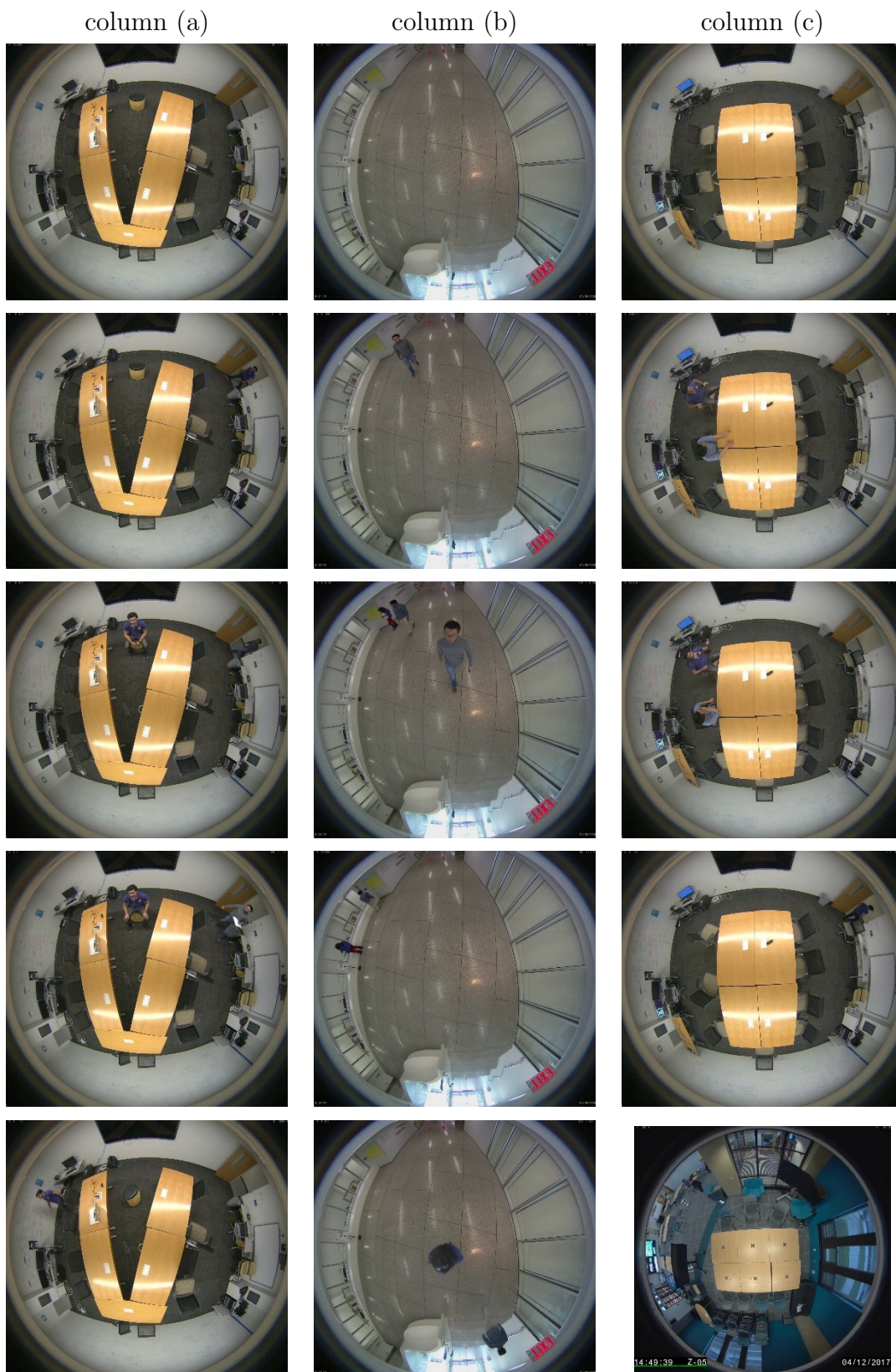





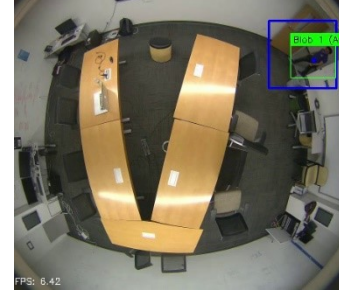
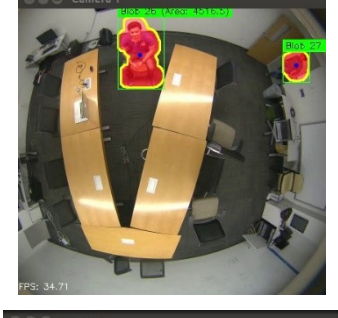
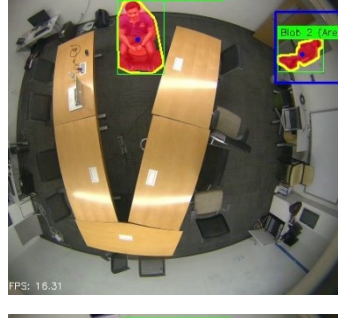
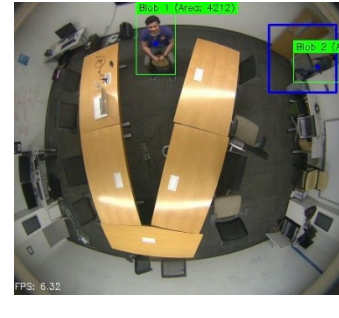



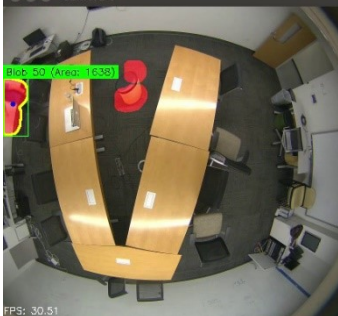
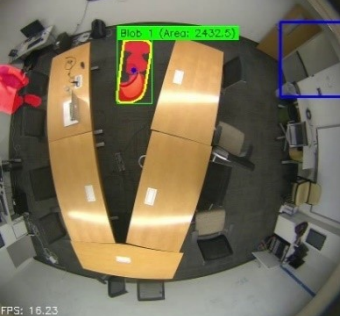





Table 4.3: Experimental results for scenario I

column (a)	column (b)	column (c)
		
		
		
		
		

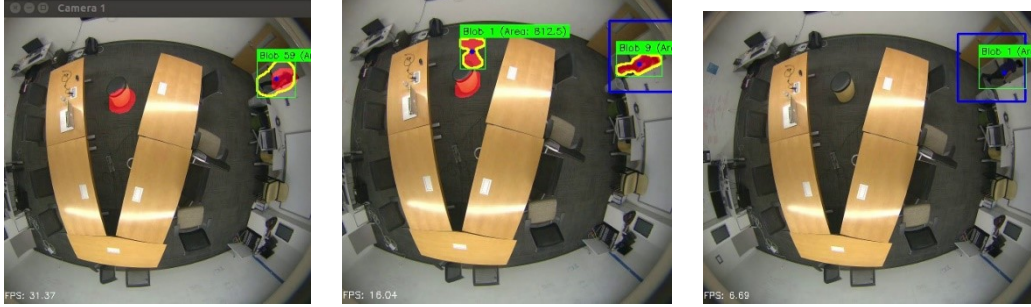


Table 4.4: Evaluation results for Scenario I

	Background Subtraction	Heuristics + A	Neural Networks + B
	(A)	(B)	(C)
True Positives	1807	2346	3618
False Positives	931	1607	237
False Negatives	2062	1517	188
Total frames with DT or GT valid	2076	2076	2076
Total GT Detections	3995	3995	3995
Precision $\langle TP/TP+FP \rangle$	0.659971	0.593473	0.938521
Recall $\langle TP/TP+FN \rangle$	0.467046	0.6073	0.950604
False Negative Rate $\langle FN/FN+TP \rangle$	0.532954	0.3927	0.0493957
F- Measure $\langle 2*precision*recall/ precision + recall \rangle$	0.546996	0.600307	0.944524
SFDA normalized Value	0.760869	0.683145	0.971712
Accuracy $\langle TP + TN / TP+FP+FN+TN \rangle$	0.3764583333	0.4288848263	0.8948800396

Table 4.5: Experimental result for scenario II

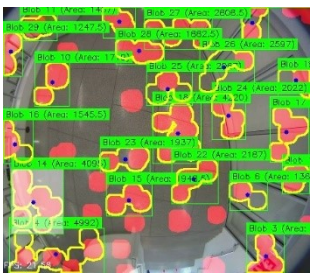
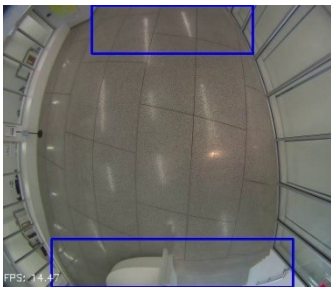
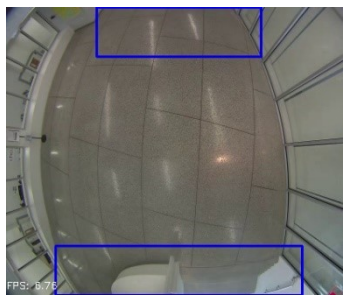
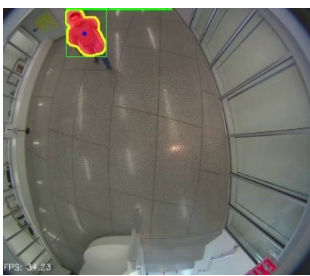
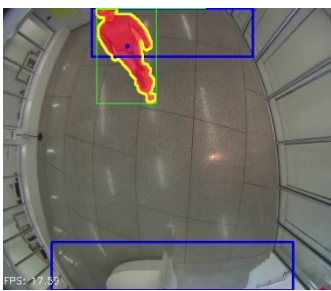
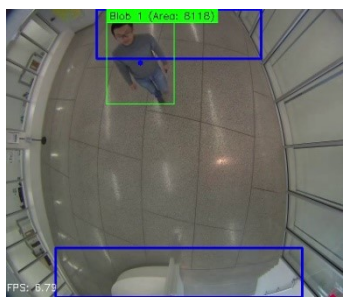

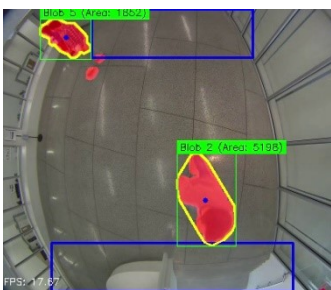
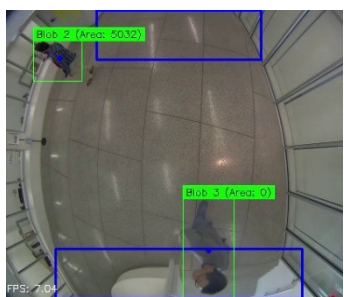
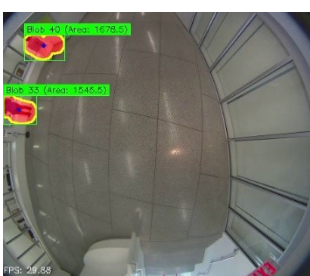
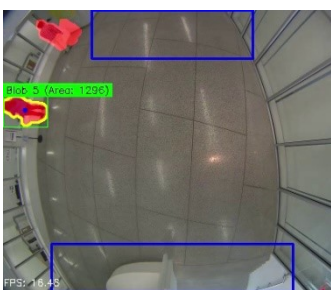


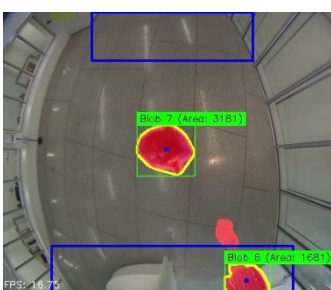
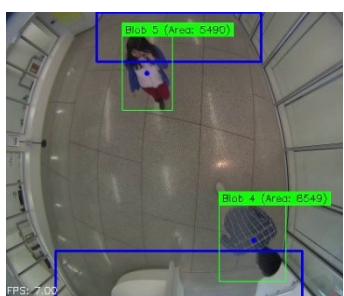
column (a)	column (b)	column (c)
		
		
		
		
		



Table 4.6: Evaluation Result for Scenario II

	Background Subtraction	Heuristics + A	Neural Networks + B
	(A)	(B)	(C)
True Positives	1630	1240	1257
False Positives	202	113	689
False Negatives	490	668	950
Total frames with DT or GT valid	1126	1126	1126
Total GT Detections	2493	2493	2493
Precision $\langle TP/TP+FP \rangle$	0.889738	0.916482	0.64594
Recall $\langle TP/TP+FN \rangle$	0.768868	0.649895	0.569551
False Negative Rate $\langle FN/FN+TP \rangle$	0.231132	0.350105	0.430449
F- Measure $\langle 2*precision*recall/ precision + recall \rangle$	0.824899	0.760503	0.605345
SFDA normalized Value	0.812964	0.907487	0.847501
Accuracy $\langle TP + TN / TP+FP+FN+TN \rangle$	0.70198105	0.61355764	0.43404696

Table 4.7: Experimental result for scenario III

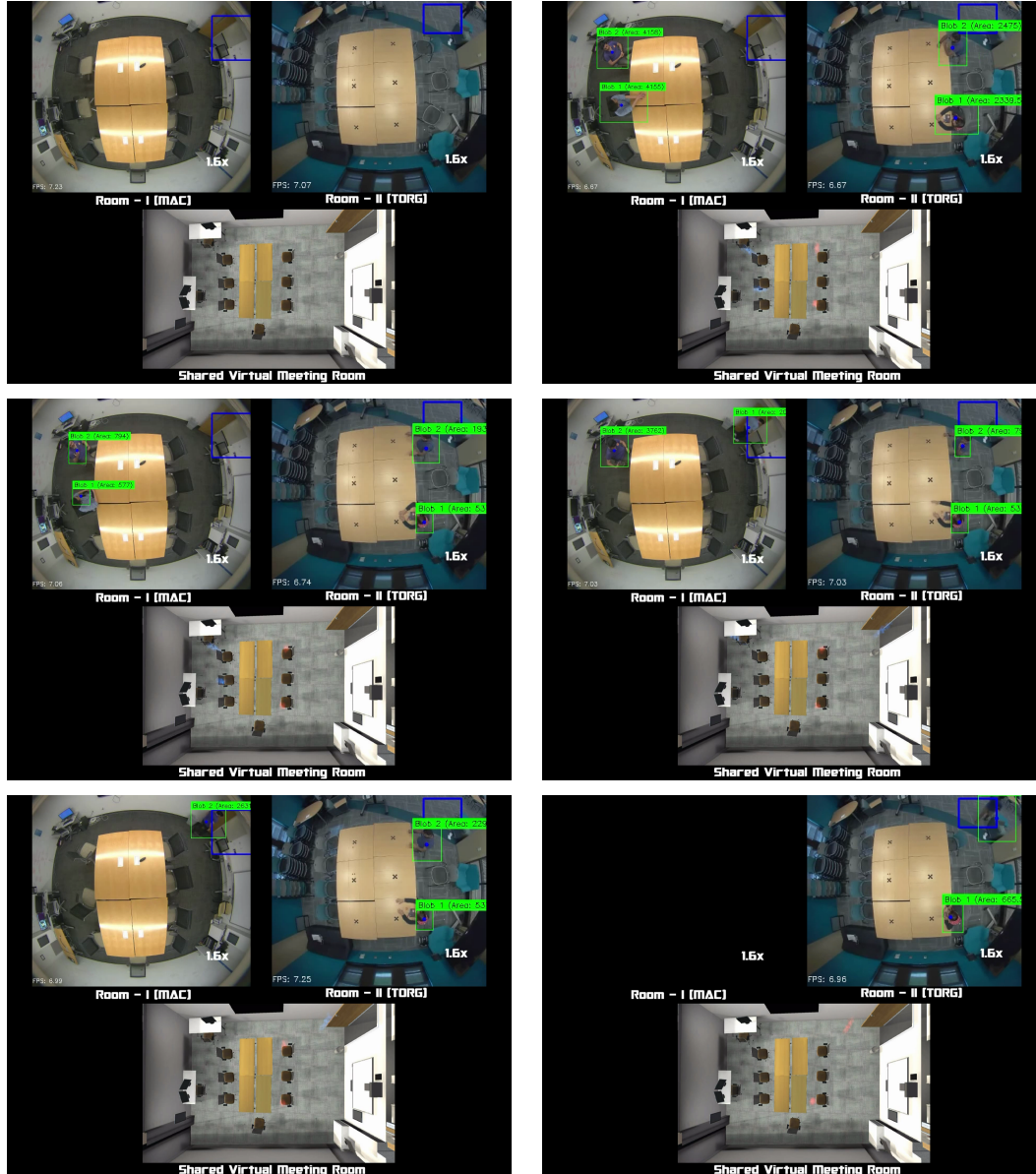


Table 4.8: Evaluation Result for Moss Art Center video sequence - Table.4.7, Room I

	Background Subtraction	Heuristics + A	Neural Networks + B
	(A)	(B)	(C)
True Positives	1542	2154	3533
False Positives	327	3245	1979
False Negatives	3414	3107	1729
Total frames with DT or GT valid	2765	2765	2765
Total GT Detections	5365	5365	5365
Precision $\langle TP/TP+FP \rangle$	0.82504	0.398963	0.640965
Recall $\langle TP/TP+FN \rangle$	0.311138	0.409428	0.671418
False Negative Rate $\langle FN/FN+TP \rangle$	0.688862	0.590572	0.328582
F- Measure $\langle 2*precision*recall/ precision + recall \rangle$	0.451868	0.404128	0.655838
SFDA normalized Value	0.388985	0.4619	0.723857
Accuracy $\langle TP + TN / TP+FP+FN+TN \rangle$	0.2918796139	0.253233012	0.4879160337

Table 4.9: Evaluation Result for Torgersen Hall video sequence - Table. (4.7), Room II

	Background Subtraction	Heuristics + A	Neural Networks + B
	(A)	(B)	(C)
True Positives	707	3837	3948
False Positives	315	2607	3059
False Negatives	5229	2116	2128
Total frames with DT or GT valid	3118	3118	3118
Total GT Detections	6258	6258	6258
Precision $\langle TP/TP+FP \rangle$	0.691781	0.595438	0.563437
Recall $\langle TP/TP+FN \rangle$	0.119104	0.644549	0.64977
False Negative Rate $\langle FN/FN+TP \rangle$	0.880896	0.355451	0.35023
F- Measure $\langle 2*precision*recall/ precision + recall \rangle$	0.203219	0.619021	0.603531
SFDA normalized Value	0.180347	0.731508	0.76649
Accuracy $\langle TP + TN / TP+FP+FN+TN \rangle$	0.1131019	0.44824766	0.43218391

Chapter 5

Collaborative Object Manipulation

In the previous chapters, we have covered how the robust combined approach based on background subtraction and YOLO [17] is useful to create a shared platform for remote presence application. This chapter covers another application of Mirror World project based on the second input device shown in Fig. 3.1. In this chapter, we will go through details about the design of collaborative object manipulation application, user study conducted to evaluate the effect of fidelity in performing collaborative object manipulation task and some of the interesting findings from this user study.

5.1 Introduction

Remote collaboration is a useful and challenging research topic, with many researches trying to innovate effective ways to create such platform. The need for remote collaboration arises due to the globalization with designers located in distant geographical locations and the multinational corporations diverse team culture. While simple audio or video conferencing technology such as Skype or Google Plus Hangout might be sufficient for communicating with families, these tools do not provide professionals with a shared task space where they can collaborate on more complex tasks, e.g. an architectural design meeting in which these experts must be able to visualize digital content at various level of details and modify the

content [99]. In addition, many of these remote collaboration tasks need an effective platform to data visualization to observe the virtual content in 3D world.

This is when Mirror World platforms comes to the aid. The concept of having Fusality between this project not only allow seamless interaction between the physical space and virtual space, but also it allows helps in delivery of spatial and gestural cues of remote collaborators in ways never thought possible, making it a suitable system for remote collaboration. The recent advancements in VR have taken significant strides with respect to the growth of the display technologies, especially head mounted displays (HMDs) and rifts. Various display and sensor technology are today available both to the academic community and the public, allowing researchers to focus on the interaction aspect of designing a VR-based user interface instead of reinventing the tracking capability or display technology.

The goal of our object manipulation task was to build the tallest possible tower with the virtual cubes in given time limit. For our trial studies, we assigned the task of assembling a tower of maximum height within a fixed time limit and evaluated participants based on the metrics of time, accuracy and satisfaction. Two (can be extended to more) people present in a virtual workbench would collaborate to achieve this common goal. We vary independent variables such as field of regard (Oculus Rift vs Screen display), complexity of target assembly (using shadow effects), and presence of physics in the virtual environment and evaluate the effectiveness of the system based on the metrics of time, accuracy, and satisfaction to replicate the target assembly.

5.2 System architecture

The overall system is made up of multiple components.

- Server side instance, with unity client running on a computer connected with Leap Motion controller.
- Client side instance on a computer connected to Leap Motion controller.

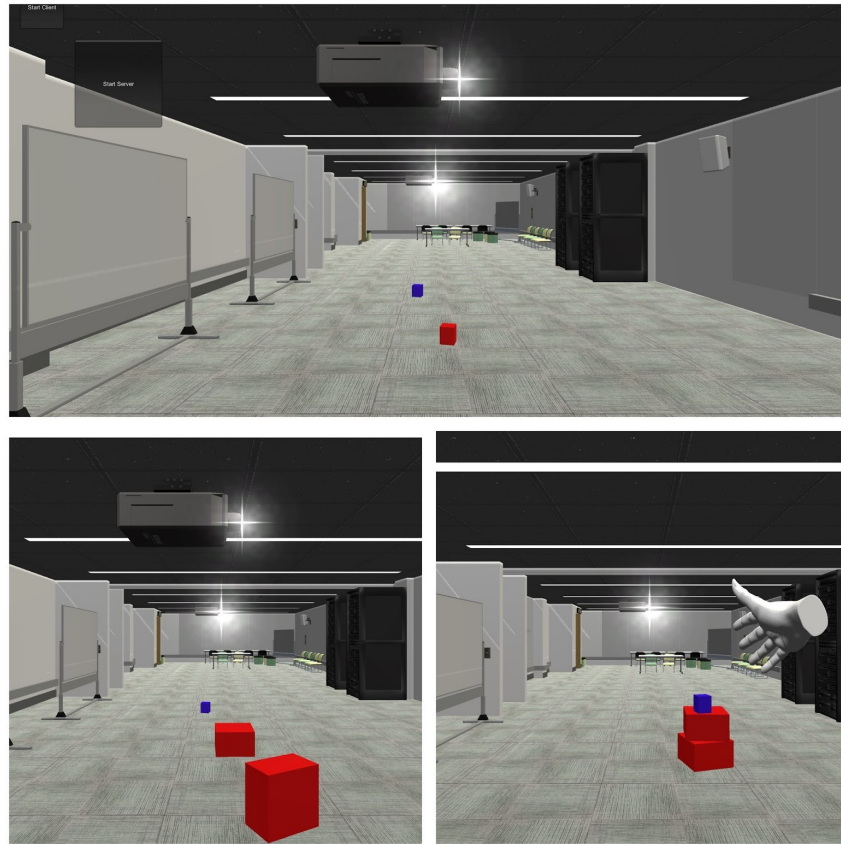


Figure 5.1: User interface for object modelling task

- Server
- Oculus Rift

The setup is shown in Fig.5.1.

5.2.1 Hardware Module Working

Initially the application is started on the two computers with one selected to act as server and other as client. The cube objects are then spawned in client and server. The Leap Motion controller client captures the hand motion over it and sends the data to unity interface which accepts the data and then translates the hand motion into virtual hand motion and in turn

moves the cube. The position of the cubes is then transferred to the server, which in turn forwards the coordinates of to the other instance of the program over the network. This act synchronizes the position of the cubes between the instances, leading to real time update and hence the collaboration. The users can also add oculus to their computers, which will enable them to view the virtual world in 3-dimensional fashion thus offering a 3-d perspective which could facilitate a better interaction with the objects in the virtual world.

5.2.2 Software Module Working

5.2.2.1 Platform

We used the same Mirror World model as shared workplace. In addition to these models, the users were classified in 2 teams based on the color of the objects. This selection was prior to starting the interaction task. The virtual cubes were generated in the MirrorWorld model, at Learning Lab at Moss Art Center which was selected as collaborative shared space between these users because it has more interaction space and multiple users can navigate in the environment conveniently.

The project involved,

- Creation of these virtual cubes based on the user input
- Adding the physics of interaction with the other objects like collision detection, rigid body etc.
- Adding the physics of gravity, which determines the nature of interaction and behavior of the object in the virtual world.
- Controlling the object with respect to Leap Motion controller's input.

5.2.2.2 Network & connectivity

Real-time collaboration between two users is a key component in this project. This involved establishing a connection between two users (systems) over the network using Web

Socket, which uses an underlying TCP connection to communicate between the server and the clients. In this project, we have created a central server component using Tornado server in Python. The Tornado server is a Python based Web framework and asynchronous library for networking that employs non-blocking network I/O and provides the framework essential for establishing a Web Socket. This server is responsible for communicating the state of objects between the collaborating user systems. The server acts as a bridge between the two users, by acquiring the state (position) of the cubes from each user environment and then communicating it to the other user. This helps in synchronizing the position of the object positions in real-time across the two users, achieving a seamless collaboration effect with changes to model getting reflected in real time.

5.3 User Study Description

We conducted a user study to test the usability and effectiveness of the system as a gestural remote collaboration system on virtual content. Subjects (13 males, 7 females) were invited to participate in a formal user test in a controlled environment in which they must perform specific tasks in pairs and provide feedback to structured questions. Each user was given 20 minutes to complete the task. Users were given a reference assembly model and asked to mimic its assembly in the virtual collaborative environment. The users were asked to use different tools for each test case and the time required to complete the task in each case was used as a performance metric. Different target models were built, varying the presence of physics and shadows. In this test case, we analyzed the response time in different work models. By employing concepts of physics, and imparting shadows to the macros in the assembling objects the difficulty level and thrill is further enhanced. Post the system interaction, the participants were then asked to fill out a post experiment survey in which they had to give ratings to the overall system performance based on different metrics. The detailed questionnaire was then used to analyze the overall system performance.

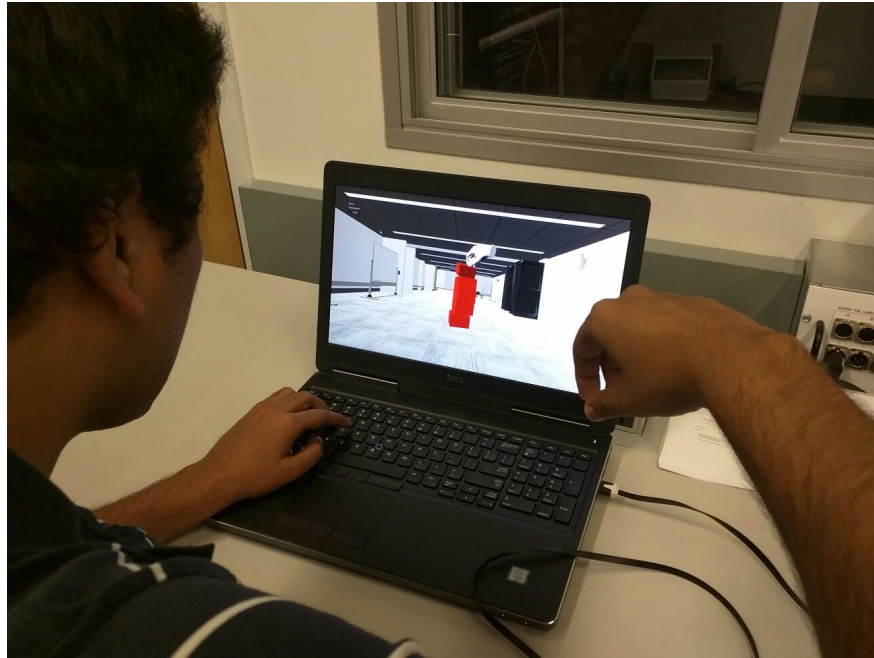


Figure 5.2: User study without Oculus



Figure 5.3: User study with Oculus

5.4 Results of the study

5.4.1 Quantitative results

In this study, we discovered the correlation between different parameters like users experience with playing games, interacting with 3D devices, 3D Design software, and Purdue Spatial Visualization Test [9] scores with their over-all performance in object manipulation task is calculated using the pearson product moment correlation coefficient in equation 5.1.

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2 \sum(y - \bar{y})^2}} \quad (5.1)$$

where, x and y are the sample means of the 2 arrays of values. The strong correlation is indicated by R value close to +1 and negative correlation is indicated by value close to -1. From the figures, it can be inferred that there is quite a strong correlation ($R = 0.587$) between the users Purdue Spatial Visualization test score and the rating given by the users for the overall experience with our system Fig. 5.1. The correlation between the users experience with 3D software or video games and the overall rating given by them is not very strong, but is still positive hence significant (Fig. 5.5, Fig. 5.6, 5.7).

Users were asked to rate their ease with using the Leap Motion controller on a scale of 1 to 7. The mean is 4.2 and standard deviation is 1.66 shown in Fig. 5.8. User ratings for the preference of Oculus for the task of assembling objects have a mean of 5.19 and standard deviation of 0.95 shown in Fig.5.9. User ratings for the preference of physics for completion of the task have a mean of 5.8 and standard deviation of 1.12 shown in Fig. 5.10. User ratings for the value of shadows for completion of the task have a mean of 4.95 and standard deviation of 0.97 shown in Fig. 5.11.

In this study, we also recorded the number of cubes assembled by the user and time taken by each user for this task. A two-by-two ANOVA was performed for cube manipulations per minute with/without Oculus and with/without Physics. The results indicate that the effect of HMD device like Oculus is significant with a p value of 0.015 and F value of 3.966. Similarly effect of physics is also statistically significant with p value of 4.186E-16 and F value of 3.966. However, the interaction between two is not statistically significant.

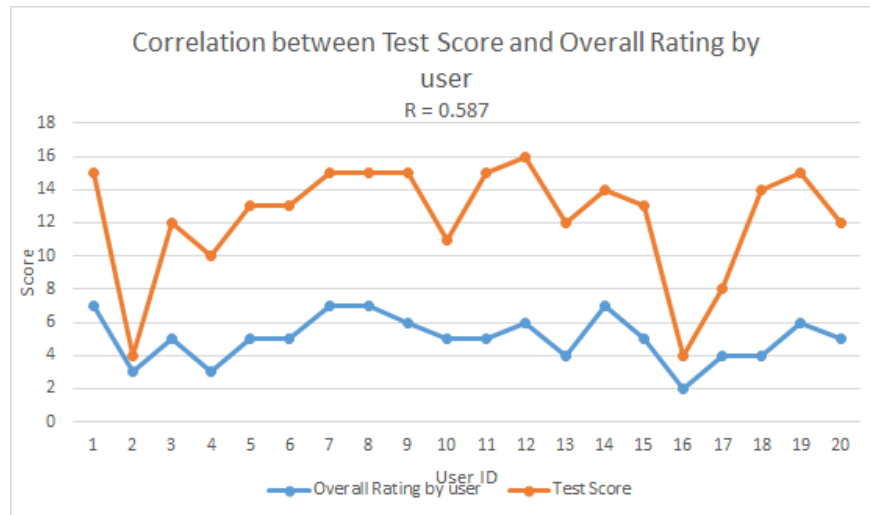


Figure 5.4: Correlation between Purdue Spatial Visualization Test [9] Score and Overall rating by User $R = 0.587$

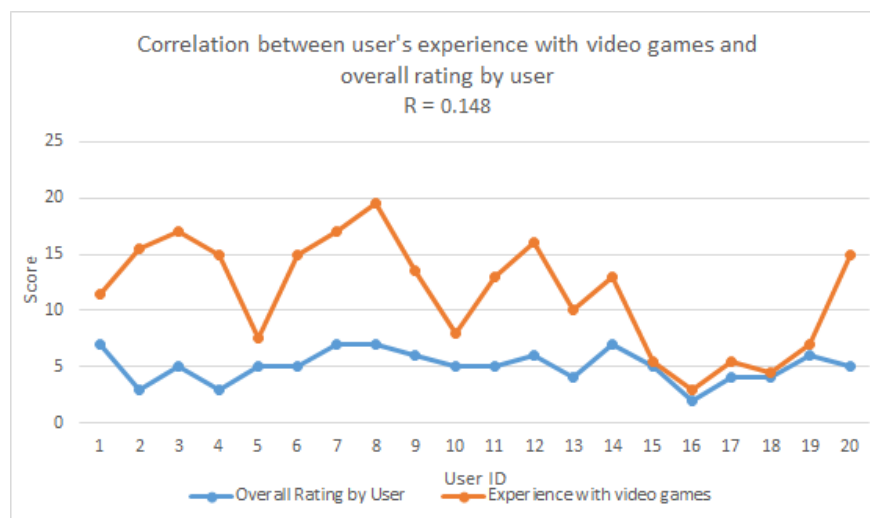


Figure 5.5: Correlation between users experience with video games and overall rating by user $R = 0.148$

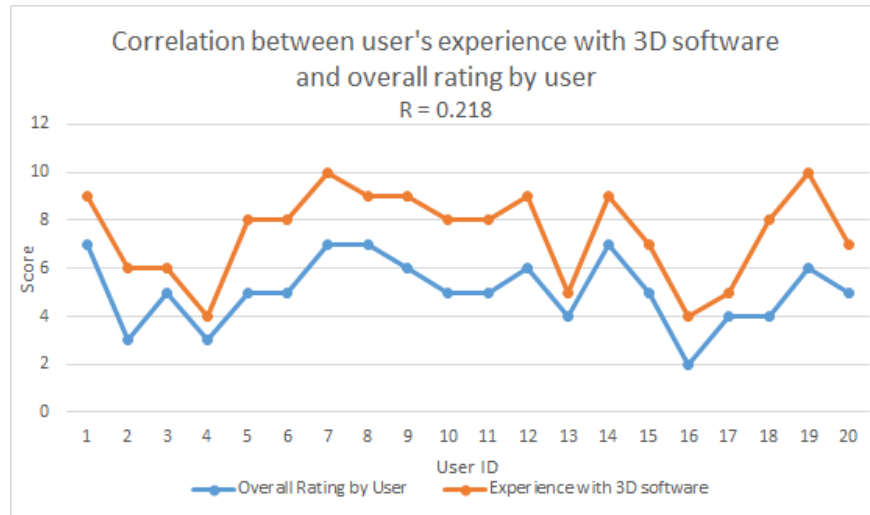


Figure 5.6: Correlation between users experience with 3D software and overall rating by user
 $R = 0.218$

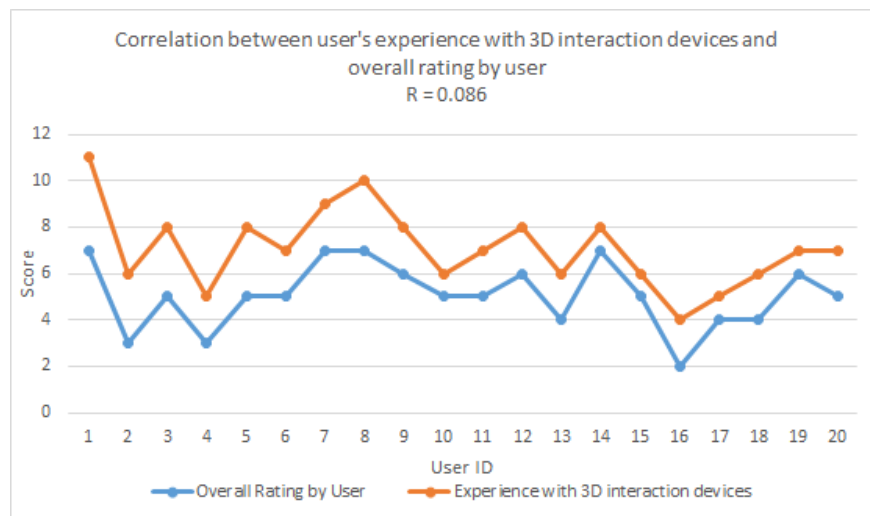


Figure 5.7: Correlation between users experience with 3D interaction devices and overall rating by user $R = 0.086$

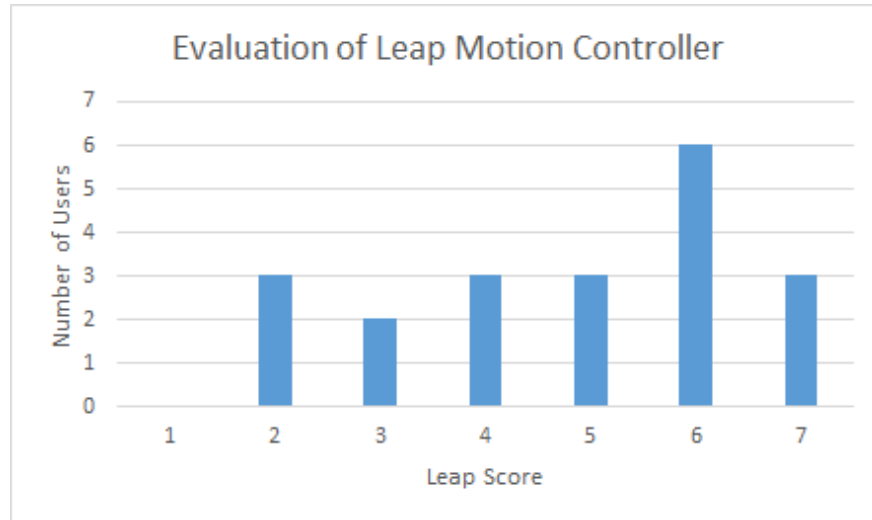


Figure 5.8: Evaluation of Leap Motion Controller

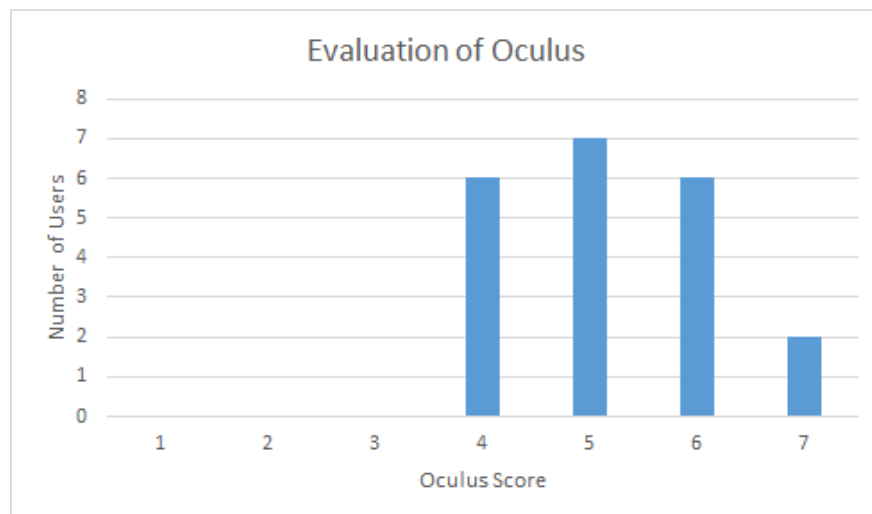


Figure 5.9: Evaluation of Oculus

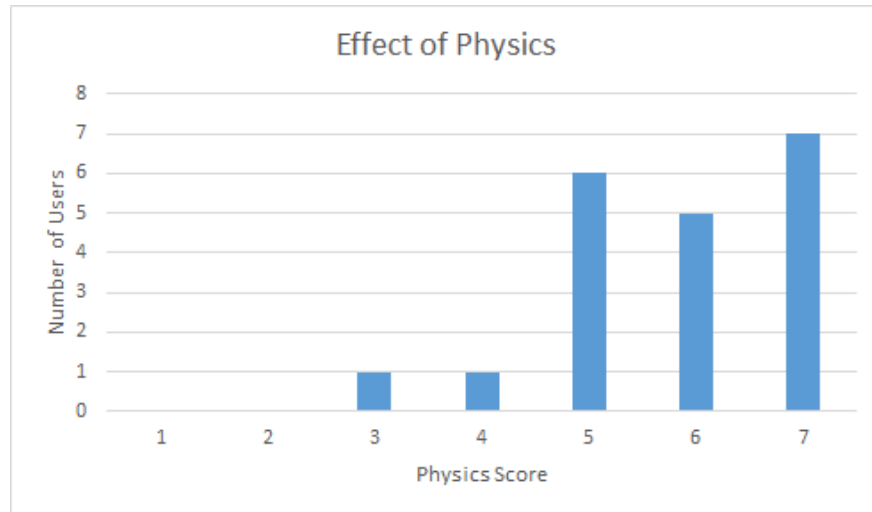


Figure 5.10: Evaluation of presence of physics

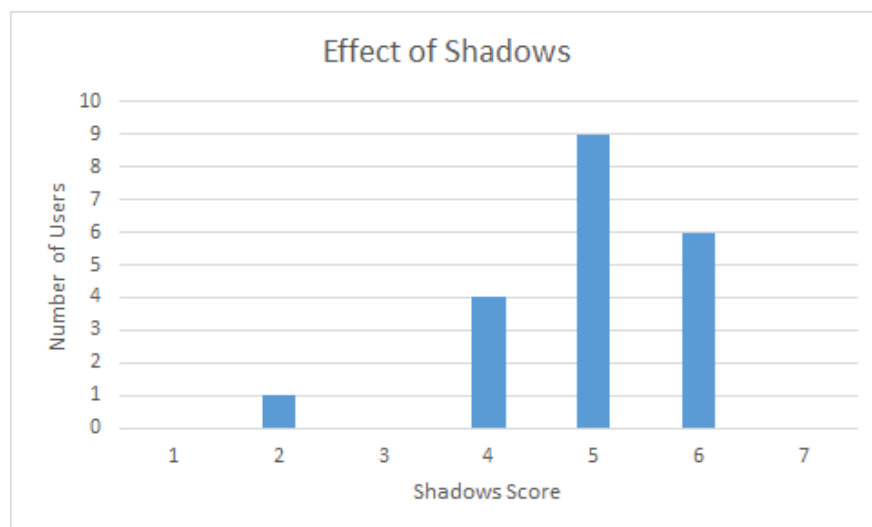


Figure 5.11: Evaluation of effect of shadows

Table 5.1: Performance analysis for tasks without Oculus

With Oculus	
With shadows	0.346 cubes/min
Without shadows	0.275 cubes/min
With Physics	0.402 cubes/min
Without physics	0.051 cubes/min

Table 5.2: Performance analysis for tasks without Oculus

With Oculus	
With shadows	0.346 cubes/min
Without shadows	0.275 cubes/min
With Physics	0.402 cubes/min
Without physics	0.051 cubes/min

Another ANOVA was evaluated over cube manipulations per minute with/without Oculus and with/without Shadows. In this result, Oculus is statistically significant with a p value of 0.002 and F value of 3.966. But the effect of shadows and the interaction between the 2 conditions is not statistically significant. We have summarized average number of cube manipulations per minute in Table. 5.1 and 5.2.

Chapter 6

Conclusions

This thesis analyzes the various issues and challenges involved in detecting the human position data using omni-directional cameras with an overhead view. There are many algorithms for detecting foreground objects based on background subtraction [20][5], HOG detectors [8], and R-CNNs [22] [88]. Some of them suffers through getting accurate results from the overhead view whereas CNN, R-CNN based detectors lags in real-time performance. The proposed system detect and track people present in building using combined approach of background subtraction and YOLO detector based on convolutional neural network [17] with heuristics based on spatial information about the physical space. The issue of stationary target was solved by detecting person using neural network detector and heuristics. An algorithm and compared against our baseline algorithms: First background subtraction (A), and second background subtraction with heuristics. Sequential Frame Detection Accuracy (*SFDA* [7]) for proposed algorithm in case scenario I is 97.17% with 89.48 % accuracy which is approximately 50% greater than baseline algorithms (A) and (B) Table 4.4. However, scenarios with moving targets (scenario II), baseline algorithms performance was comparable and plain background subtraction method performed slightly better than proposed algorithm. The algorithm was scaled to support network cameras installed across multiple buildings to create remote presence application.

Different systems were studied to overview about the applications designed for telepresence

and remote collaboration. Most of systems were using stereo cameras, Kinect sensors, network cameras etc. These systems are limits the user space over constraint environment like laboratory, classroom or offices. The proposed system uses omni-directional cameras installed in the building to create a telepresence application. People moving in building can see their reflection as an avatar in the tablets mounted on the side-walls. Users in one physical space (Moss Art Center) can interact with the remote participant (Torgersen Hall) as different avatar in the virtual model. Remote participants can present in virtual model through parallel system installed in Torgersen Hall or they can connect to the virtual model over web X3D clients.

Thesis also presents the collaborative object manipulation platform designed using Leap Motion controller. The proposed system uses Oculus Rift as a display device and gesture input using Leap Motion controller. In alternative approach, the Kinect device was used as the hand gesture recognition tool but the large space requirement of the Kinect made Leap Motion controller a better choice for this task. The qualitative results showed overall user satisfaction and ease of handling and working with the system. The quantitative results exhibited positive correlation between Purdue spatial visualization test [9] score and overall performance which implies that user with good 3D spatial sense performed better in assembling the target model. The other observation was that gender bias did not affect the user performance but experience with 3D software, gaming does seem to correlate with superior user performance.

Chapter 7

Future Work

To improve the performance of the system, as a future work, system can be improved in four primary aspects of the Mirror Worlds project [4] as follows:

- Current system uses neural network trained on ImageNet [89] and MS COCO [100] datasets. The performance of the detector can be improved by training on dataset collected for this system [4].
- Tracking algorithm currently relies on detection results. It can be improved by using feature based tracking algorithms [101]
- The avatars representing people presence in building can animate the actions performed by users. Actions like walking, orientation, change in position information can be extracted based on additional heuristics and blob data
- Collaborative object manipulation application can be designed using HoloLens device to create mix reality experience.

Bibliography

- [1] Maribeth Back, Don Kimber, Eleanor Rieffel, Anthony Dunnigan, Bee Liew, Sagar Gattepally, Jonathan Foote, Jun Shingu, and Jim Vaughan. The virtual chocolate factory: Building a real world mixed-reality system for industrial collaboration and control. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1160–1165.
- [2] Don Kimber, Jim Vaughan, and Eleanor Rieffel. Augmented perception through mirror worlds. In *Proceedings of the 2nd Augmented Human International Conference*, page 36. ACM, 2011.
- [3] Don Kimber, Jun Shingu, Jim Vaughan, David Arendash, David Lee, and Maribeth Back. Through the looking glass: mirror worlds for augmented awareness & capability. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1269–1270. ACM, 2012.
- [4] MirrorWorlds, project website. <http://icat.vt.edu/mirrorworlds/>. Accessed: 2017-04-16.
- [5] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27:773–780, may 2006.
- [6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *ArXiv e-prints*, June 2015.

- [7] Vasant Manohar, Padmanabhan Soundararajan, Harish Raju, Dmitry Goldgof, Rangachar Kasturi, and John Garofolo. Performance evaluation of object detection and tracking in video. *Computer Vision–ACCV 2006*, pages 151–161, 2006.
- [8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- [9] Roland Guay. *Purdue Spatial Vizualization Test*. Educational testing service, 1976.
- [10] David W Schloerb. A quantitative measure of telepresence. *Presence: Teleoperators & Virtual Environments*, 4(1):64–80, 1995.
- [11] Jim Hollan and Scott Stornetta. Beyond being there. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 119–125. ACM, 1992.
- [12] Rick Fry and Gene F Smith. The effects of feedback and eye contact on performance of a digit-coding task. *The Journal of Social Psychology*, 96(1):145–146, 1975.
- [13] Gwyneth Doherty-Sneddon, Anne Anderson, Claire O’malley, Steve Langton, Simon Garrod, and Vicki Bruce. Face-to-face and video-mediated communication: A comparison of dialogue structure and task performance. *Journal of experimental psychology applied*, 3:105–125, 1997.
- [14] Tom DeFanti, Dan Sandin, Maxine Brown, Dave Pape, Josephine Anstey, Mike Bogucki, Greg Dawe, Andy Johnson, and Thomas S Huang. Technologies for virtual reality/tele-immersion applications: issues of research in image display and global networking. In *Frontiers of Human-Centered Computing, Online Communities and Virtual Environments*, pages 137–159. Springer, 2001.
- [15] Gregorij Kurillo, Allen Y Yang, and Ruzena Bajcsy. 3d telepresence for reducing transportation costs. 2016.
- [16] Gregorij Kurillo and Ruzena Bajcsy. 3d teleimmersion for collaboration and interaction of geographically distributed users. *Virtual Reality*, 17(1):29–43, 2013.

- [17] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.
- [18] Luis M Fuentes and Sergio A Velastin. People tracking in surveillance applications. *Image and Vision Computing*, 24(11):1165–1171, 2006.
- [19] Nicholas F. Polys, Benjamin Knapp, Matthew Bock, Christina Lidwin, Dane Webster, Nathan Waggoner, and Ivica Bukvic. Fusality: An open framework for cross-platform mirror world installations. In *Proceedings of the 20th International Conference on 3D Web Technology*, Web3D 15, pages 171–179. Proceedings of the 20th International Conference on 3D Web Technology, ACM, 2015.
- [20] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.
- [21] Tracking-MATALB, algorithm description. <https://www.mathworks.com/help/vision/ug/multiple-object-tracking.html>. Accessed: 2017-01-11.
- [22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587. Proceedings of the IEEE conference on computer vision and pattern recognition, 2014.
- [23] Ramesh Jain and H-H Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):206–214, 1979.
- [24] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfnder: Real-time tracking of the human body. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):780–785, 1997.
- [25] Xiang Gao, Terrance E Boulton, Frans Coetzee, and Visvanathan Ramesh. Error analysis of background adaption. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 503–510. IEEE, 2000.

- [26] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, page 252 Vol. 2. Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), 1999.
- [27] Ahmed M. Elgammal, David Harwood, and Larry S. Davis. Non-parametric model for background subtraction. In *Proceedings of the 6th European Conference on Computer Vision-Part II, ECCV 00*, pages 751–767. Proceedings of the 6th European Conference on Computer Vision-Part II, Springer-Verlag, 2000.
- [28] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 255–261. IEEE, 1999.
- [29] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International journal of computer vision*, 38(1):15–33, 2000.
- [30] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE transactions on pattern analysis and machine intelligence*, 23(4):349–361, 2001.
- [31] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [32] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, volume 12, pages 296–301, 1995.
- [33] William T Freeman, Ken-ichi Tanaka, Jun Ohta, and Kazuo Kyuma. Computer vision for computer games. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 100–105. IEEE, 1996.

- [34] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32:1627–1645, 2010.
- [35] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.
- [36] MachineLearningClass, class website. <https://filebox.ece.vt.edu/~f16ece5424/>. Accessed: 2016-10-27.
- [37] David López Vilariño, Victor M Brea, Diego Cabello, and JM Pardo. Discrete-time cnn for image segmentation by active contours. *Pattern Recognition Letters*, 19(8):721–734, 1998.
- [38] Isaak Kavasidis, Simone Palazzo, Roberto Di Salvo, Daniela Giordano, and Concetto Spampinato. A semi-automatic tool for detection and tracking ground truth generation in videos. In *Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications*, page 6. Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications, 2012.
- [39] Jia Deng, Alex Berg, Sanjeev Satheesh, H Su, Aditya Khosla, and L Fei-Fei. Imagenet large scale visual recognition competition 2012 (ilsvrc2012), 2012.
- [40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [41] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

- [42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.
- [43] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013.
- [44] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.
- [45] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 17–24, 2013.
- [46] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
- [47] Shai Avidan. Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence*, 29(2), 2007.
- [48] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 260–267. IEEE, 2006.
- [49] Bastian Leibe, Konrad Schindler, and Luc Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

- [50] Kenji Okuma, Ali Taleghani, Nando de Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. *Computer Vision-ECCV 2004*, pages 28–39, 2004.
- [51] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.
- [52] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.
- [53] Weixin Chen and John L Barron. High accuracy optical flow method based on a theory for warping: 3d extension. In *International Conference Image Analysis and Recognition*, pages 250–262. Springer, 2010.
- [54] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [55] Jerome Berclaz, Francois Fleuret, and Pascal Fua. Robust people tracking with global trajectory optimization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 744–750. IEEE, 2006.
- [56] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision*, pages 788–801. Springer, 2008.
- [57] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1683–1698, 2008.
- [58] Yuan Li, Chang Huang, and Ram Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2953–2960. IEEE, 2009.

- [59] AG Amitha Perera, Chukka Srinivas, Anthony Hoogs, Glen Brooksby, and Wensheng Hu. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 666–673. IEEE, 2006.
- [60] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [61] David Gelernter. *Mirror Worlds: or: The Day Software Puts the Universe in a Shoe-box... How it Will Happen and What it Will Mean*. Oxford University Press, 1993.
- [62] Joshua Lifton and Joseph A Paradiso. Dual reality: Merging the real and virtual. In *International Conference on Facets of Virtual Environments*, pages 12–28. Springer, 2009.
- [63] Saul Greenberg and Chester Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209–218. ACM, 2001.
- [64] TYZX. stereo camera description. <http://tyzx.com/products/DeepSeaG2.html>.
- [65] Joanna C Coelho and Fons J Verbeek. Pointing task evaluation of leap motion controller in 3d virtual environment. *Creating the Difference*, 78, 2014.
- [66] Mukund Raj, Sarah H Creem-Regehr, Kristina M Rand, Jeanine K Stefanucci, and William B Thompson. Kinect based 3d object manipulation on a desktop display. In *Proceedings of the ACM Symposium on Applied Perception*, pages 99–102. ACM, 2012.
- [67] Stefano Scheggi, Leonardo Meli, Claudio Pacchierotti, and Domenico Prattichizzo. Touch the virtual reality: using the leap motion controller for hand tracking and wearable tactile devices for immersive haptic rendering. In *ACM SIGGRAPH 2015 Posters*, page 31. ACM, 2015.
- [68] Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss, and Andrew Wilson. Holodesk: direct 3d interactions with a situated see-through display. In *Proceedings of*

- the SIGCHI Conference on Human Factors in Computing Systems*, pages 2421–2430. ACM, 2012.
- [69] Vivotek, ip camera. <http://www.vivotek.com/fe8174>. Accessed: 2015-09-20.
 - [70] oncam, ip camera. <https://www.oncamgrandeye.com/security-systems/evolution-05-concealed-camera.html>. Accessed: 2015-09-20.
 - [71] vtwiki, virginia tech wiki. <https://webapps.es.vt.edu/confluence/display/ICATVT/Vivotek+network+camera+Installation>. Accessed: 2016-02-16.
 - [72] vrml, introduction page. <https://www.w3.org/MarkUp/VRML/>. Accessed: 2017-02-10.
 - [73] x3d, introduction page. <http://www.web3d.org/getting-started-x3d>. Accessed: 2017-02-10.
 - [74] Dan Tilden, Ankit Singh, Nicholas F. Polys, and Peter Sforza. Multimedia mashups for mirror worlds. In *Proceedings of the 16th International Conference on 3D Web Technology*, pages 155–164. Proceedings of the 16th International Conference on 3D Web Technology, 2011.
 - [75] Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zllner. X3dom: A dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D 09, pages 127–135. Proceedings of the 14th International Conference on 3D Web Technology, ACM, 2009.
 - [76] Andrea Prati, Ivana Mikic, Mohan M Trivedi, and Rita Cucchiara. Detecting moving shadows: algorithms and evaluation. *IEEE transactions on pattern analysis and machine intelligence*, 25(7):918–923, 2003.
 - [77] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.
 - [78] M. and M. Wand Jones. *Kernel Smoothing*. Chapman and Hall, London., 1995.

- [79] Peter Hall, Tien Chung Hu, and J. S. Marron. Improved variable window kernel estimates of probability densities. *The Annals of Statistics*, 23:1–10, 1995.
- [80] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30:32–46, 1985.
- [81] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.
- [82] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP09*), pages 331–340. International Conference on Computer Vision Theory and Application VISSAPP’09), INSTICC Press, 2009.
- [83] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3:209–226, 1977.
- [84] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1:79–83, 1982.
- [85] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. Advances in neural information processing systems, 2012.
- [86] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [87] R. G. J. S. Shaoqing Ren, Kaiming He, and R. C. N. N. Faster. Towards real-time object detection with region proposal networks, arxiv preprint. *arXiv preprint arXiv:1506.01497*.
- [88] Ross Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*. The IEEE International Conference on Computer Vision (ICCV), dec 2015.

- [89] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.
- [90] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.
- [91] Margrit Betke and Zheng Wu. Data association for multi-object visual tracking. *Synthesis Lectures on Computer Vision*, 6:1–120, 2016.
- [92] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82:35–45, 1960.
- [93] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowd-sourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.
- [94] Carl Vondrick, Deva Ramanan, and Donald Patterson. *Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces*. Springer Berlin Heidelberg, 2010.
- [95] Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba. Labelme video: Building a video database with human annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1451–1458. IEEE, 2009.
- [96] Amol Ambardekar, Mircea Nicolescu, and Sergiu Dascalu. Ground truth verification tool (gtvt) for video surveillance systems. In *Advances in Computer-Human Interactions, 2009. ACHI09. Second International Conferences on*, pages 354–359. Advances in Computer-Human Interactions, 2009. ACHI’09. Second International Conferences on, 2009.
- [97] Simone Bianco, Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. An interactive tool for manual, semi-automatic and automatic video annotation. *Computer*

- Vision and Image Understanding*, 131:88–99, 2015. Special section: Large Scale Data-Driven Evaluation in Computer Vision.
- [98] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.
 - [99] Travis B Faas. An examination of social presence in video conferencing vs. an augmented reality conferencing application. 2010.
 - [100] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C.L. Zitnick, and P. Dollr. Microsoft coco: Common objects in context. *ArXiv e-prints*, may 2014.
 - [101] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 798–805. IEEE, 2006.