Key Aspects of Integrating Augmented Reality Tools into Peer-to-Peer Remote Laboratory User Interfaces

Ananda Maiti, Alexander Kist, Mark Smith
School of Mechanical and Electrical Engineering, University of Southern Queensland, Australia
anandamaiti@live.com, kist@ieee.org, andrew.maxwell@usq.edu.au

***Abstract*— Augmented Reality (AR) is the process of overlaying meaningful information in a video stream for creating an enriched visual experience for users. Peer-to-Peer (P2P) Remote Access Laboratories (RAL) are systems where individual users or makers can build their experimental setups, program the rig and then share it with other users. This enables them to get design experience along with gaining knowledge about the particular experiment in question and potentially collaborate on design experiences. This paper focuses on the integration of AR tools with a P2P RAL system. The challenges of the P2P RAL system are that makers are to be provided with a generalized set of tools that they can use to create the experiments and program them and the AR tools have to be generic and applicable for multiple experiments. The corresponding issues discussed are the applications of AR in RAL, the levels of AR in context of RAL and their effect on the learning tools, the nature of the P2P platform, the challenges of integrating AR in the generic P2P programming platform and their solutions. The methods of implementing the AR tools in the P2P Platform are also presented.**

***Keywords*— augmentated reality; remote laboratories; e-learning; digital image processing; object recognition**

I. INTRODUCTION (*Heading 1*)

Remote Access Laboratories (RAL) are online environments that allow users to communicate with remote instruments over the internet. The users employ a User Interface (UI) that provides information regarding the experiments and inputs and outputs. The UI is sometimes referred to as a client. For an effective learning experience, UIs must be well designed. Several technologies have been used to create appropriate UI for RAL including specialized versions for desktops [12] and mobiles [13]. A Peer-to-Peer (P2P) RAL enables individual users (or *makers*) to create RAL experiment and add them to the system [10]. The P2P RAL aims to create a set of tools that provide homogeneity in the design of user interface, its interaction with the users and instrument control. This means the maker (or experiment creators) must be provided with web based tools to create the program and UI for the experiment they design and host [5].

Augmented Reality (AR) is the technology to embed media information in video streams to create a rich interactive user Interface (UI). The embedded AR components can be text or highly complex graphics. The AR technology reacts to the surrounding environment i.e. responds with AR components depending upon the visual inputs to the system. The AR components must update in real time by recognizing the input video frames' contents and processing it according to a pre-determined logic to produce the AR components. Other common inputs apart from static objects are gestures form the users themselves which is also part of the environment. AR systems can also take conventional inputs such as mouse and keyboards.

This work is supported through the Australian Government's Digital Futures CRN program.

AR is used in many areas of science and technology including computer games for recreational purposes, sports and entertainment, navigations and tourism. AR has also been used in education [6]. In this paper, the aspects of integrating AR into RAL are discussed. A P2P RAL creates the challenge that the experiments must all be provided with a set of tools that are useful for all types of experiments. Thus a common set of requirements and conditions are considered. Four different levels of relationships between the real and virtual components in an AR application have been identified depending upon their activeness. Two different solutions of has been proposed as a part of the P2P RAL system to deal with these cases, super-imposing animated and interactive objects on video streams and identifying and tagging objects to corresponding sensor values. These solutions in form of AR tools can be used for multiple experiments independently designed by different makers.

The rest of the paper is organized as follows: Section 2 discusses the current status of AR and it's applications in Education and in particular RAL. Section 3 discusses the P2P RAL system and identifies the conditions and contestants of AR in RAL context. The SNAP programming platform [5] and the AR tools are presented in Section 4. Section 5 discusses the implementation methods of the tools described.

II. RELATED WORK

A. Augmented Reality

Augmented Reality can be perceived in multiple ways. Most commonly it is defined as a mixed environment that blends digital information with real world objects in a meaningful way [1]. The amount of real world entities in the environment should be more than the overlay information for it to be AR [7].

There are different classes of AR environment based on how immersive they are. One common form of devices includes a head mounted displays system and possibly hand gloves with feedback [3]. These are fully immersive environments that enable users to experience whole of the reality environment with augmented features. It also allows more accurate interaction with the AR environment. Fully

Fig. 1. The Epson Moverio BT-200 smart glass. Smart glasses are capable of providing real time fully immersive AR features.

immersive AR is achieved by using a wearable device such as smart glasses or head mounted displays. These devices have cameras mounted on them which are capable of running applications to process the video which is the visual area of the user. The view of the user is then enhanced with overlaid information.

The other type is desktop AR [4] which only covers partial portion of the surrounding of the user, in particular what can be shown in the desktop screen. The view is limited and interaction with the environment happens with regular input devices e.g. mouse, keyboard, etc. While full immersive AR is more attractive and advantageous, they have many problems:

- Expensive: The hardware required is expensive for being a commonly available tool to be used for educational purposes.
- Requires high precision to recreate the augmented feature. It requires expertize to setup and maintain the system. Prone to errors [14].
- It works by augmenting the local environment with virtual objects. This by itself takes considerable cost, processing power and technology. It is more difficult and unnecessary to recreate a remote real environment completely and then augment it with virtual objects. The ability to view remote real environment is much reduced and only a fixed set of views are available through cameras. Thus these video streams can be projected directly onto screens.

With immersive AR, it is difficult to obtain a generalized environment to create a number of interfaces from a single platform. This causes disparity between the interfaces between different systems with little in common. Also, the immersive AR is not always necessary for good educational outcome [1].

B. *AR in Education*

AR in education mainly aims to provide a rich educational experience. Such systems usually concentrate on the desktop AR or mobile devices. Traditionally, some systems use markers for identifying the location in the real world stream to be replaced with the augmented information of objects as well as a unique identifier for what to display. The augmented objects are stored in a database, against a unique identifier and reproduced when the desired marker with that identifier appears in the screen. This also requires accurate computer vision techniques to correctly identify the marker and the encoded identifier within it.

This type of technology helps in understanding operation and models of the objects that are available in-place with real world learning materials. They present the users with a quick in-depth augmented multimedia experience during their interaction with real world environment.

C. *RAL and AR*

Augmented Reality features have been added to RAL experiments before [6-8]. Usually, the AR is desktop type and mostly the augmentation is overlaid virtual components such as switches that can be manipulated by the user. In [7] the virtual elements on FPGA boards, users can remotely interact with the real and virtual devices. The real devices are viewed through a camera video feedback. This approach gives a very realistic environment, as majority of what the users see as part of the user interface is the real object i.e. the FPGA Board enabling to view all the changes and event clearly. Only small portions of the video feed back is overlaid with other information and graphics that takes users inputs.

The main limitations in the broader context of P2P RAL are that these examples are designed specifically for one experiment. Moreover, the co-ordinates of the virtual objects are directly tied to the co-ordinates of the hardware in the video feedback. Any change in the hardware orientation may require change in the AR setup as well. As part of a P2P RAL, a common web-based instrumentation platform is used by multiple users with different hardware setups. Thus the P2P RAL systems must identify which objects in the real environment need to be supplemented. This enables makers to specify certain objects in the video feedback and associate the virtual components with them within the online environment.

III. Augmented Reality and RAL

This section describes the application areas of AR in RAL, and types of AR and constraints of applying them.

A. *AR Application Areas in RAL*

In general Augmented Reality (AR) can be used in many ways [1] but the most common approach is to draw virtual objects onto the real world video feed and then updating them. The augmented reality in RAL can be used for two purposes:

- *Induced visibility*: In certain experiments some objects/entities may not be visible to the camera. For e.g. magnetic fields that attract magnetic materials generated and studied by using different electromagnets [9]. This entities which are part of the experiment, may be implanted into correct positions by using animations. This involves re-drawing certain objects such as arrows over the region to indicate the presence and orientation of the entities.
- *Overlaid Information*: Another set of objects that needs to be presented is text information relating to certain real objects in the video. It is best to draw the text onto the video feedback close to the associated object. To do this however, the objects must be identified and tracked in real-time during the experiment. Overlaid text information must be updated in real time as well to reflect the change in the state of the object.

B. *Levels of AR*

In the current context, AR is the process of overlaying virtual objects including scalar images or vector animations or both onto the video feedback. The video feedback has a definite frame rate and resolution and thus a fixed number of pixels (P) for each frame. For the AR, a pixel p in the feedback may contain either real-object or virtual object (or maybe fractionally both). Thus two measurements can be defined -

1. Virtual Pixels (P_V): the average number of pixels that relate to a virtual object. Then, ΔP_V and ΔR_V are two parameters that signifies average change in the P_V and R_V over time in the video feedback where R_V is the matrix representing the position of the virtual pixels.
2. Real Pixels (P_r): the number of pixels that relate to real objects. ΔP_r and ΔR_r signifies average change in the P_r and R_r over time in the video feedback where R_r is the matrix representing the positon of the real pixels.

This allows for different degrees of virtual and real objects to be blended. In the P2P RAL, the AR may be implemented by having

Case 1. More virtual components with more active behavior than that of the real objects. i.e.

$$P_V > P_r \text{ and } (\Delta P_V)(\Delta R_V) > (\Delta P_r)(\Delta R_r)$$

Case 2. More virtual components than the real objects but less active in behavior than the real objects. i.e.

$$P_V > P_r \text{ but } (\Delta P_V)(\Delta R_V) < (\Delta P_r)(\Delta R_r)$$

Case 3. Lesser virtual components than the real objects but more or equal active in behavior than the real objects.

$$P_V \leq P_r \text{ but } (\Delta P_V)(\Delta R_V) \geq (\Delta P_r)(\Delta R_r)$$

Case 4. Lesser virtual components than the real objects. i.e.

$$P_V < P_r \text{ and } (\Delta P_V)(\Delta R_V) < (\Delta P_r)(\Delta R_r)$$

The first scenario is in the space of augmented virtuality [2] (not in scope of this paper) where both virtual visibility and associated information are high and the real objects themselves do not change their orientation much. In the second scenario real objects change their orientation more often compared to (or equally to) the induced visibility/information. In the 3rd scenario the user have less virtual components and the real world objects, both can be equally active. In the 4th scenario, the users' interact largely with the real components and only supporting information is displayed as visible information.

C. P2P RAL Challenges

P2P RAL is a system where individual *makers* can make their experimental setups, program them and then share it with other users (see Fig 2) [5]. This enables them to get design experience along with gaining knowledge about the particular experiment in question and potentially collaborate on design experiences which are vital for STEM education [10]. This however, poses some challenges:

1. Individuals should spend more time on the experimental problem and designing the setup. The overhead activities of automating the setup and creating the UI should be minimal.
2. Users are spread over a large geographic are with varying hardware and setup plans even for the same experiment.

To address these issues, a Micro-Controller Unit (MCU) based architecture is used. The MCUs have a set of ports that can be connected to sensors and actuators to control. This is supported by an online platform to connect the MCUs with a corresponding UI. The connection is established by the RLMS. It also stores the UI files and any other relevant data regarding an experiment created by a maker. The UI contains all components that are presented to the user of the experiment. This UI takes inputs from the users, process them according to a control logic specified by the maker and them communicates with the MCU. Thus all AR capabilities are embedded in this online platform. Makers can use AR tools to incorporate virtual objects onto the UI along with a video feedback from the corresponding Camera.

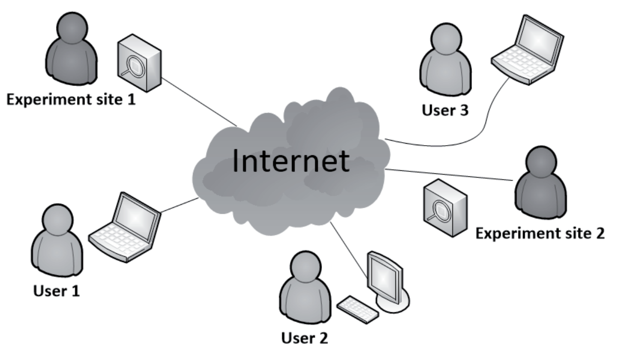


Fig. 2. The P2P RAL architecture.

IV. AR Framework for the P2P RAL

This section presents the AR framework of the P2P RAL environment [5].

A. P2P RAL Experiemnts Design

Creating an experiment involves the following steps:

1. *Constructing the rig*: To construct the rig, MCUs such as Arduinos, LEGO Mindstorms etc. are used. The MCU forms the core of the experimental rig which may be built with everyday objects. All peripheral units are sensor and actuators are connected to the MCU through unique ports. The ports can be identified by unique numbers. The MCU process the incoming user commands and makes changes in the rig accordingly. It also reads values from sensors and returns them to the UI if required.

The MCUs are connected to a network router called RALfieBox which opens a VPN connection to the P2P RAL system [15]. The VPN system enables communication between any RALfieBox and any other device on the internet enabling an end-to-end connection between the users' devices (PC or mobile devices) to the makers experiment MCUs.

2. *Programming the Rig*: The P2P RAL system is based around a SNAP platform [5] for programming the rig. Originally, SNAP is a graphical language created with HTML5 to work in web browsers (http://snap.berkeley.edu/). It is aimed at young students to learn computer programming. For the P2P RAL, it has been modified to communicate with MCUs and operate on the internet. The SNAP platform allows the makers to create a *control logic* (CL) and the UI for an experiment. The UI takes inputs from the users and displays the output. The SNAP platform combines the users inputs with the *control logic* to determine any action or command incorporating the ports that is sent to the MCU.

The SNAP programming platform is provided through a cloud environment. It is used by both the makers of

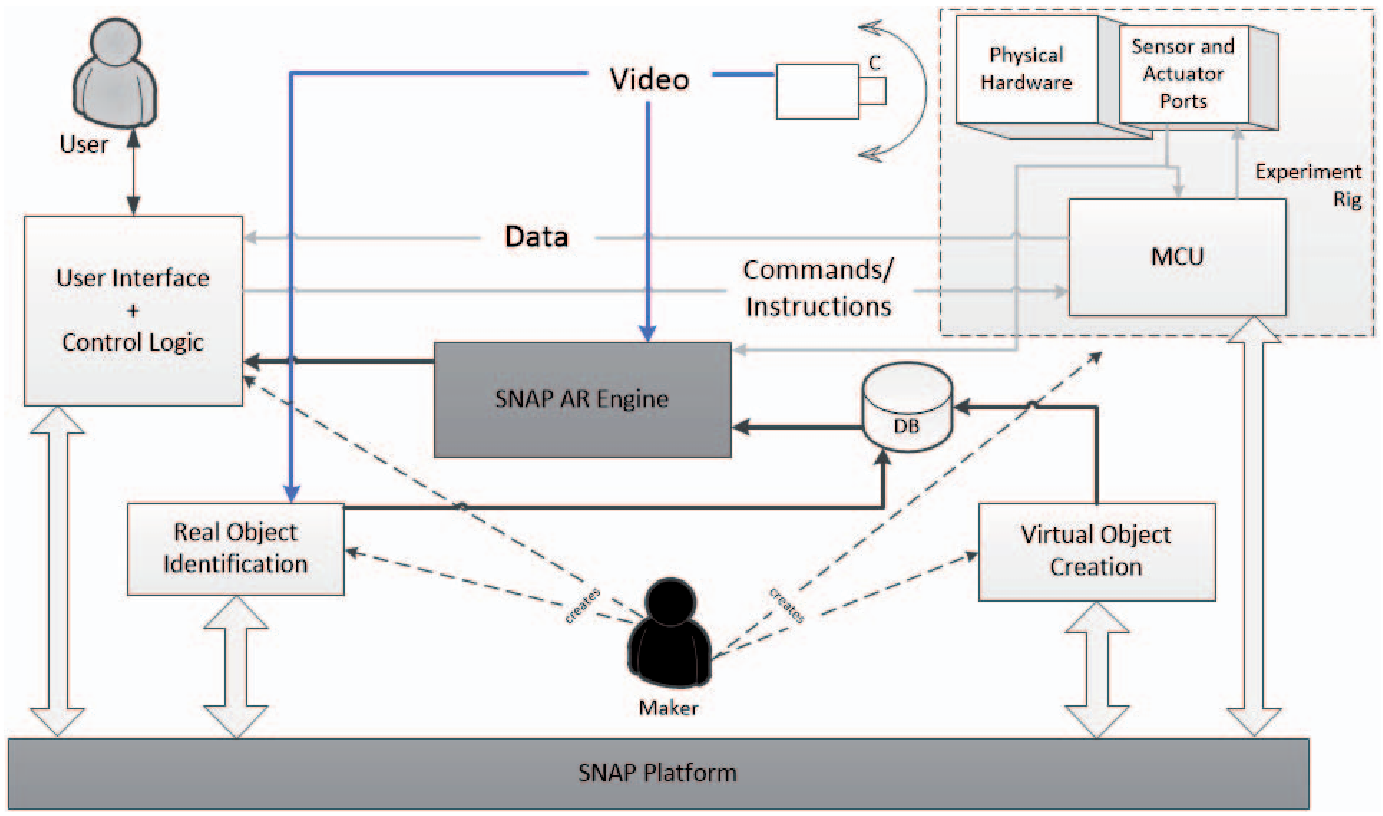


Fig. 3. The SNAP environment and the experiment rig

experiments to create the program logic etc. for an experiment and the users who run the experiment for learning purposes.

3. *Publishing it to the web*: The makers can create and re-design the rigs and the CL/UI as they want. Finally when they are satisfied with the rig, the experiment can be made public. The users can then access the UI to run the experiment, but cannot change it.

The video feedback is through an IP Camera connected with the RALfieBox that in turn connects the UI on the users' devices to the video stream along with the MCU. It may be also possible to use a webcam. The camera view is usually static in coordinates and same resolution but the maker may enable it to be controlled by the user depending upon the requirements of the experiment. Each camera is uniquely identified and can be controlled from the SNAP Platform. In case of AR based experiments, it is desirable to keep the cameras static but resolution may be altered.

B. *Integrating AR in the P2P System*

The role of AR comes into the Programming part of the rig. The AR information includes two type of procedure:

- *Virtual Object Creation (VOC)*: The makers can create animations in the SNAP environment and these needs to be aligned correctly to the video feedback. This can be achieved by ensuring that the virtual objects and all of their re-orientations are with the bounds of the real objects coo-ordinates in the video feedback.
- *Real Object Identification (ROI)*: As mentioned earlier, makers are not expected to create markers [8] for AR objects. It is also not possible as the SNAP system does not know what the maker wants in the AR. Thus any real object that needs to be augmented with virtual objects must be identified by the SNAP system in the video feedback. Once the desired objects are identified, they may be associated with corresponding virtual objects or overlaid text information.

The maker has to perform these tasks as a part of building the rig which is supported by the SNAP platform. The UI and CPL is also created in the SNAP Platform. A separate AR engine (also part of the SNAP platform) can combine the stored Virtual objects and corresponding Virtual objects/components into the video feedback during run-time for any user.

Figure 3 shows the SNAP system with the various AR related component. There are two separate streams that feed to the UI - the resultant data and the SNAP AR frames. The resultant data is the data obtained from the experimental rig i.e. sensor data of actuator success or failure data. The SNAP AR Engines generates the frames for the UI that shows the video frame received from the experimental setup modified with the AR components. The video feedback is received through an IP Camera as a MJPEG stream in the SNAP which is further procced according to the experiment and the corresponding objects saved in the database. The maker is responsible for both ROI and VOC both of which are optional for a given experiment. The objects identified and their associated media by ROI or created by VOC and their activities are stored in a database alongside the experiment. This database is used by the

SNAP AR engine. The SNAP AR tools run on top of the SNAP execution engine that processes the users' program and communicates with the experiment.

C. P2P AR Applications

For P2P RAL, the cases 2, 3 and 4 can be addressed as follows.

1. The *Cases 2* and *3* can be handled by super-imposing the desired virtual objects on to the camera feedback. This allows for the camera feedback to directly display the real objects without any alteration. This will work only if the amount of virtual objects pixels is less than or equal to the number of pixels for the real objects. This does not work well if ultimate the number of virtual object pixels are greater than that of real objects as a large number of pixels are required to be re-drawn in each frame of the video, reducing system performance. It also increases complexity of real-virtual object pairing and the ways to store them in the database and display them farther affecting performance and experience.
2. The *Case 4* is displaying information associated with certain components of the rig. The virtual components are not special objects but only text that are updated in real time. In the rigs, each actuator will result in change of orientation. This change may be associated with a certain component in the rig and thus a corresponding sensor may be able to read the changed values. These values can be shown in real time over or near the component in the video feedback.

This helps makers in:

- understanding key components of the design and identify any weakness
- quickly associate sensor value to any object with the need of displaying them explicitly. This saves screen space which is very important for mobile devices.

For the user, obvious the augmented components helps in identifying and understanding the changed in the experiment easily without having to look into detailed UI reports [12].

V. Implementations

This section discusses the methods to implement the two solutions to the *Cases 2, 3* and *4* as generic tools for the online SNAP platform in conjunction with a P2P RAL.

A. Super-Imposing the Camera View

The SNAP platform has a designated area of screen that is called stage. The stage is where all the objects of animation and other output data are displayed. The simplest form of AR is to super-imposing the cameras view below the stage. This is done by connecting the stage's background directly to the camera stream. The camera steam may be resized to any size and placed at any position on the stage or a full screen mode can be applied.

Makers must include a command to start the AR. If the AR is not started then the SNAP environment behaves like a typical non-AR setup. Once the AR mode is started, the camera is visible. Then the makers can include any object they wish on the stage that will appear on top of the video stream. The makers can make precise movements according to the underlying changes in the camera feedback.

Fig 5 shows an example of this type of AR. The experiment activity concerned is a traffic light system as shown in Fig 4. The rig has some LEDs that must go 'on' and 'off' and virtual objects are the cars which must stop and move according to the LEDs status. The LEDs are connected to the ports on the MCU and controlled with commands/instructions from the SNAP UI to the MCU. The video feedback shows the LEDs and the background roads etc. and the cars' movements are controlled through SNAP depending upon the data received from MCU. This type of AR presents two kinds of issues:

1. *Stability*: The camera view is assumed to be static. The problem may arise if and when the camera view changes due to the camera getting moved accidentally, in which case, the changed view can be detected and the makers can be notified to duly put the camera back in place. However, if the AR objects are very precisely programmed, they may require re-calibration of coordinates to ensure correct UI interactions. There is no object identification procedure to re-align the virtual objects accordingly.
2. *Response time*: The response time is the time taken to retrieve any data/video from the maker node to the user node. On the internet this may be high. The animation frame rate will be typically faster than the frames from the video, thus it will create a lag in user interaction if every frame of the stage is attached to a new video frame. For this purpose, the video is handled by a separate process that runs parallel to the snap execution platform running the virtual

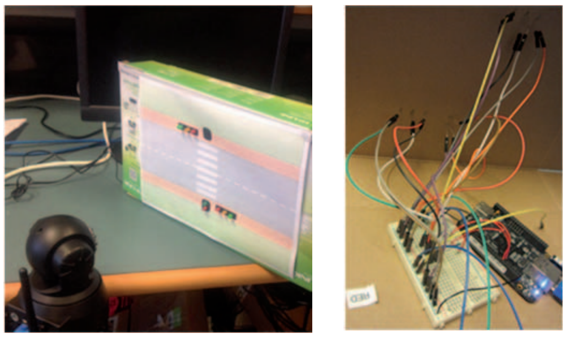


Fig. 4. A P2P RAL experiment using everyday objects and MCU

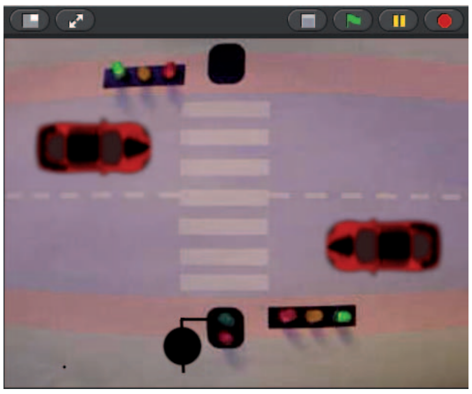


Fig. 5. A traffic light example in SNAP with real LEDs and virtual cars

objects activity. Whenever the video frame is retrieved, the stage background is updated accordingly. Thus the users' interaction with the UI components remains normal.

A second problem is the difference between the arrival rate of data and the SNAP animation frame rate. Due to response time, the data may arrive at a later time than the relevant frame where the data was supposed to have any effect. Thus, all SNAP execution including animation is suspended when a message or instruction a is issued from the SNAP to the MCU at time T_a. During this period the virtual objects do not move or operate thus creating a paused state until the data is received. With higher latencies, the number of paused states will increase in an interval of time, thus affecting quality of experience. But the data and SNAP animation will remain synchronized thus not affecting the learning objectives.

The last problem is the arrival time difference between the data and video stream. Typically, the command and sensor data exchanged between the nodes is very small and delivered at faster rate than the video as well. This causes the problem of de-synchronization between the video frame and the virtual objects. Thus, the SNAP execution engine is paused until a new video frame T_a^v is received after receiving a new data T_a^d after an instruction a is issued. With stable internet conditions, there will be least effect on the performance and interaction of the users. Thus the paused time after an instruction a is issued to the experimental rig from the user interface is:

$$Paused\ Time(a) = \max\{\, T_a^v - T_a\,,\ \ T_a^d\ -\ T_a\}$$

B. *Object Identification and Tagging*

The second approach is the solution to Case 4 where the objective is to identify individual objects in the video stream and tag them. The objects cannot be mapped to any fixed global database in the SNAP as there is no limit to what the users can use to create the rig. Thus the SNAP platform must be able to store these additional components alongside the control logic program for every experiment.

The following steps are describes the process of identification and tagging:

- The SNAP AR engine identifies objects that change position (or shape) over time. This process is ROI.
- A record of the desired objects is created for the particular experiment based upon physical properties of the objects i.e. color, contour, size or even an image of the object. The record is stored in the database for the experiment alongside the control logic.
- Once these objects are recorded, the makers can attach a sensor's value to the object, which is also stored in the corresponding record.

In any rig, the actuators move causing a change in the rig's position. The magnitude of the change in the position can be measured by a sensor. For e.g. in Fig 6 (b) showing a pendulum experiment, if the user has to drop the ball, then the corresponding actuator is rotated by a certain degree. This ball then changes position in the video stream. The length of the drop is a function of the rotation. This change in the balls position can be display in real time as augmented texts pointing towards the ball.

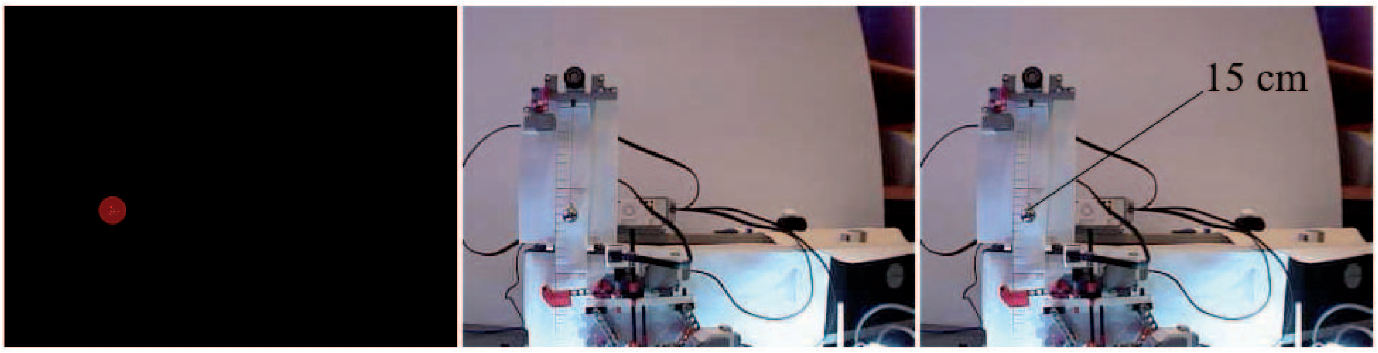
The *maker* is able to attach the sensor values x as a function *f(x)* which is constantly updated on the screen. The maker can also designate a particular area of the screen where the text is displayed. This should ideally be a space that does not have any meaningful object and the text should not overlap such objects. However, in certain cases where, the rig has massive change in position, no such suitable space on the screen may be available for the entire duration of the experiment. The SNAP AR engine must determine a suitable space to put the text. The user is also able to switch on and off the AR components to make them visible or invisible.

A prototype ROI mechanism has been developed in P2P RAL - SNAP as follows:

Step1. The initial image of a video stream when a session starts is stored as the background image (B).

Step2. Once any object moves in image F_i, it is isolated by subtracting F_i' = F_i - B. A residue of the object is left in B, which is identified using subsequent frames, as the residue will always remain static. The pixels of the object residue in B are replaced with the corresponding pixels in the current frame F_i.

Step3. A clustering mechanism is used to remove noise and get the actual objects.in the frame F'_i. The clustering mechanism takes into account the potential radius (as specified by the maker) of the target object that needs to be



(a) (b) (c)

Fig. 6. The pendulum Experiment. (a) The difference in frames to identify the moving object (i.e. the ball) (b) The original video feedback of the pendulum experiment (c) The final video feedback with the sensor value as shown to users.

tagged. Thus any object that is larger than the size is automatically put of the list. Figure 6(a) depicts an identified object.

Step4. The makers can then select the object(s) that need to be stored permanently and will be used for AR.

Step5. The maker then uses a SNAP block to associate a sensor value with the desired object and also mention its x, y coordinates on the stage.

Step6. Each object is stored in a database and marked with a unique identifier. The object does not need to be identified in real world as what it actually is, but only matched relatively in each experiment session.

It may be noted that the actual algorithms to realize each part can be implemented in multiple ways. For example, the DBSCAN algorithm [11] is used for creating clusters of right size.

When the users run the experiment, the AR module checks if any of the objects, stored in the database while creating the experiment is in the frame. If there is any such object, then the corresponding, sensor values are shown if AR tools are activated. Figure 6(c) shows the final output of the Object Identification and Tagging (OIT) process where the ball is tagged with the value of the sensor measuring its height. This image frame is placed in the stage of the SNAP environment.

The function of entire AR module in SNAP to create each frame of the video feedback to the user works in two steps (see Figure 7):

1. First, the video feedback is analyzed and for each frame, the objects are identifies and tagged according to the makers' selection and function.
2. Second, the virtual objects created by the makers are then placed in the video feedback.

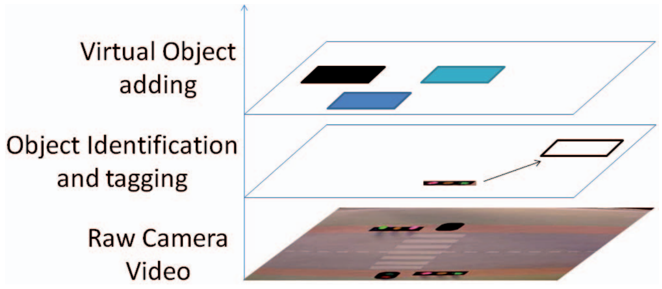


Fig. 7. The layers of AR components

The prototype system for ROI (or its implementation - OIT) is successful in principle to provide a generic tool to create AR interfaces for multiple experiments using the steps described earlier. But it is not able to support all types of experiments. The major performance shortcomings that can be refined are:

i. The JavaScript based SNAP environment runs in a web-browser. The OIT AR tools increase the CPU rate to more than 33% on a 2.5GHz, Intel i5 processor using the web browser Firefox version 41. This indicates that AR tools require considerable computational power which may be available on mobile devices.

ii. In *Step 3*, a clustering algorithm the DBSCAN algorithm is used which has 2 inputs ε and p where ε signifies the radius around a point and p is the minimum number of points (or pixels) around a given point. The resultant clusters will have each point in the cluster surrounded by a minimum of p clusters within a radius of ε. This is an ideal way to determine objects and reduce noise in the video input. However, this also adds to parameters that need to be altered to an extent to identify the desired objects correctly. Further improved implementations of this have to either automatically adjust this or the users choose the desirable values.

iii. In *Step 4*, the objects properties - average color for red, blue and green along with a range of minimum and maximum heights and widths of the detected clusters are stored. While this is sufficient to identify small and mobile objects with uniform color, it may fail on some scenarios with larger objects.

C. Limitations and Future Work

One of the limitations of the implementations described here includes the need for static camera positions. Ambient lighting changes can also affect the outcome. It will create larger differences between the background (B) and any subsequent frames. Thus the experiments with AR tools must be setup in a well-defined environment. Further work will look into minimizing the effect of response time on the performance of AR tools.

The OIT in this work assumes that there are fewer moving or changing components compared to static objects in an experiment view. At the moment issues such as occlusion are not addressed, i.e. when an object is covered by other objects. The OIT described here can be used only for moving or largely changing objects visible to the camera.

CONCLUSIONS

A peer-to-peer remote laboratory architecture incorporating augmented reality tools has been discussed. The P2P RAL must provide generic tools for all makers to create a variety of experiments. AR tools are based on the activity level of the virtual and real components in the experiment video. Such tools can help users of the experiments to quickly identify the changing parameters of the experiment and help makers get acquitted with the relationship of those parameters and the rig operation. In short, augmented reality tools can help users and makers to recognize the most important learning concepts in the experiment. It can highlight important data and help user to understand the experiment. The proposed methods are also applicable to any RAL experiment even outside P2P RAL.

REFERENCES

[1] H.-K. Wu, S. W.-Y. Lee, H.-Y. Chang, and J.-C. Liang, "Current status, opportunities and challenges of augmented reality in education," Computers & Education, vol. 62, pp. 41-49, 3// 2013.

[2] S. Ternier, R. Klemke, et. al., "ARLearn: Augmented Reality Meets Augmented Virtuality", Journal of Universal Computer Science, vol. 18, no. 15, 2012, 2143-2164.

[3] H. Kaufmann and D. Schmalstieg, "Mathematics and geometry education with collaborative augmented reality," Computers & Graphics, vol. 27, pp. 339-345, 6// 2003.

[4] J. Camba, M. Contero, and G. Salvador-Herranz, "Desktop vs. mobile: A comparative study of augmented reality systems for engineering visualizations in education," in Frontiers in Education Conference (FIE), 2014 IEEE, 2014, pp. 1-8.

[5] A. Maiti, A. D. Maxwell, A. A. Kist, and L. Orwin, "Joining the Game and the Experiment in Peer-to-Peer Remote Laboratories for STEM Education," presented at the 2015 3rd Experiment@ International Conference (exp.at'15), Portugal, 2015., pp. 213-218.

[6] A. Cardoso, et al. C. Marques, et al., "Online experimentation: Experiment@Portugal 2012," in Remote Engineering and Virtual Instrumentation (REV), 2014 11th Intl. Conf. on, 2014, pp. 303-308.

[7] J. M. Andujar, A. Mejias, and M. A. Marquez, "Augmented Reality for the Improvement of Remote Laboratories: An Augmented Remote Laboratory," Education, IEEE Trans. on, vol. 54, pp. 492-500, 2011.

[8] S. Abu Shanab, S. Odeh, R. Hodrob, and M. Anabtawi, "Augmented reality internet labs versus hands-on and virtual labs: A comparative study," in Interactive Mobile and Computer Aided Learning (IMCL), 2012 International Conference on, 2012, pp. 17-21.

[9] M. B. Ibáñez, Á. Di Serio, D. Villarán, and C. Delgado Kloos, "Experimenting with electromagnetism using augmented reality: Impact on flow student experience and educational effectiveness," Computers & Education, vol. 71, pp. 1-13, 2// 2014.

[10] A. Maiti, A. D. Maxwell, A. A. Kist, and L. Orwin, "Integrating enquiry-based learning pedagogies and remote access laboratory for STEM education," in EDUCON, 2014 IEEE, 2014, pp. 706-712.

[11] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," SIGKDD Explor. Newsl., vol. 4, pp. 65-75, 2002.

[12] Z. Nedic, J. Machotka, and A. Nafalski, "Remote laboratories versus virtual and real laboratories," in Frontiers in Education, 2003. FIE 2003 33rd Annual, 2003, pp. T3E-1-T3E-6 Vol.1.

[13] J. Garcia-Zubia, D. Lopez-de-Ipina, and P. Orduna, "Mobile Devices and Remote Labs in Engineering Education," in ICALT '08. Eighth IEEE International Conference on, 2008, pp. 620-622.

[14] D.W.F. van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations", The International Journal of Virtual Reality, 2010, 9(2):1-20.

[15] A. A. Kist, et al., "Overlay network architectures for peer-to-peer Remote Access Laboratories," in REV 2014, pp. 274-280.