Augmented Reality and Augmented Telepresence Visualization for Industrial Automation

Tyson Stolarski

Du Huynh
School of Computer Science and Software Engineering

Apolon Ivankovic
CEED Client: Sentient Computing

Abstract

A major aspect of industrial automation software is to clearly and effectively relay important information of the industrial process to the operator in a timely manner. Traditionally this has been done via two dimensional diagram based User Interfaces (UIs) with a high amount of context specific symbols and shapes that only have meaning to the engineers and operators familiar with the graphical standard. A new visualization approach utilizes the live camera feeds of modern industrial processes, augmented with the large amount of operational data available, to provide effective operations monitoring and control. This project produced a prototype augmented reality system for this very purpose, combining video feeds from a wide array of supported IP cameras, and overlaying data from a multitude of possible data sources.

1. Introduction

Augmented Reality (AR) is a general purpose term used for any view of reality where elements of that view are augmented with virtual imagery. Its uses span simple everyday applications; e.g. television broadcasts of swimming events use a virtual line to show the position of the world record holder at that time; to complex, advanced systems; e.g. the Head Mounted Display systems of modern military aircraft that overlay flight data for the pilot, based on where the pilot is looking in the cockpit (F-35 Joint Strike Fighter Program n.d).

MacroView is a Supervisory Control and Data Acquisition (SCADA) product developed by Sentient Computing that is currently being upgraded in a number of core areas. This upgrade includes improvements/additions to the back-end server infrastructure, and a series of new front-end visualizations. These additions include a new product line known as MVX (MacroView eXtreme), which provides a virtual 3D environment to view and control real time data sources from Process Control and SCADA systems. An extension to this product line is a new augmented reality visualization system – instead of a completely virtual environment, it uses real-time camera feeds to display the physical process and surrounding environment, and is then augmented with Process Control and SCADA system data.

Historically, SCADA system visualization and controls were done via 2D UIs created by process control engineers using software specific to that SCADA system. These UIs often consist of basic, 2D representations of the industrial process being monitored and controlled, using process control standards developed in the early 90s. While these UIs may make sense to the engineers that designed them, there is often a large understanding gap between them.
and the operators using them on a day to day basis. In some applications, it is also beneficial for the operator to see the physical environment where the industrial process is occurring, even though they are not physically at the location. This is usually achieved via camera feeds; however, camera feeds are always supplemented with standard UIs to provide the operator with all the data that is not visible to the naked eye, and to provide control inputs to the process. Operators are generally familiar with the physical system and have an instant understanding of what they are looking at when they see a camera feed, compared to a 2D abstraction.

This project addresses this problem via an augmented reality visualization prototype. This prototype allows for camera feeds to be configured and overlaid with important data visualizations based on the camera’s current position and orientation, and the data points configured in the SCADA system’s database. This allows an operator to quickly identify the industrial process via its physical appearance, and also quickly obtain important system data overlaid on the same visualization. This prototype visualization system is not a replacement for all 2D visualizations; instead, it is another possible visualization client to be used where applicable.

2. Project Design

The design of the augmented reality visualization software spanned multiple tiers of the SCADA software stack, and had constraints due to its coupling with the MVX product. Although it was always considered a prototype system, early design meetings with Sentient indicated that for some subsystems there would be significant benefits if existing MVX functionality was completed and then leveraged by this project. To bring this prototype to a functioning level, systems were required on both the server and client side:

1. A server system was needed to provide configuration for the available camera feeds and their location/orientation/available actuation, and to configure what industrial control data was available for each camera feed, as well as where that data came from.
2. A server-client protocol needed to be chosen that allowed for the transfer of the complex types of data configured on the server (the standard range of basic data types, video feeds, location/orientation data types, and also the relationships between camera feeds and the data they can be augmented with).
3. The visualization client that performed the actual work of displaying the camera feed to the user and augmenting with the relevant data visualizations.

2.1 MVX Object Server

The MVX Object Server is the server system planned for the whole MVX product range. At a fundamental level, it exposes a tree of objects to connected clients, and allows clients to consume the data contained within this object tree (The term ‘object’ is used here in the software sense, where it is a means of organizing data and program code so as to implement some form of automated software system). The data contained within each object can be defined directly in the Object Server, or it can be pulled from an external data source (e.g. traditional SCADA system, Video server, etc). This Object Server system was formulated at the start of the MVX product, as a solution to the lack of complex data available to industrial automation systems – traditionally industrial automation systems organize real time data as ‘points’ or ‘tags’, and then follow up with a naming convention to provide structure. By organizing these thousands of data points into a meaningful object-oriented structure, we believe we can get significantly more information to the industrial automation system clients.
This relationship based data structure is important for the augmented reality visualization client as it needs to know the relationships between camera feeds and position/orientation data, and also how each industrial control data point relates to a real world position. It was therefore decided that part of this project would consist of adding to the current MVX Object Server implementation to bring it up to a functional level required for use by this project. This would eventually have to be done for the remainder of the MVX product anyway, and therefore it was decided that it would be beneficial to get this aspect done properly now, rather than a throw-away prototype server. This constraint meant the server side implementation would be built on top of existing MacroView serialization and communication libraries, with existing implementations in both C++ and .NET (C#). Due to time constraints only the .NET server libraries were chosen. This restricted the server to a Windows Operation System, but for this prototype system that is sufficient.

ISA 95 is an international standard for the integration of enterprise and control systems (ISA European Office 2010). It was decided to standardize on hierarchy model concepts to organize the object oriented data structure for the MVX Object server. This standard helps to provide a common terminology and software model for the interfaces between enterprise and control system levels.

![Class Diagram](image_url)

**Figure 1** Class diagram of the ISA-95 compliant object model basis that all MVX Object Server data is organized into (Generated from source code).

### 2.2 Metaserver Protocol

The decision to utilize the MVX Object Server as the configuration server constrained the project to also use the existing server-client communications protocol already defined for the
Object Server. The existing MacroView .NET communication Application Programming Interfaces (APIs) utilize a TCP/IP protocol known as the Metaserver Protocol developed for the original MacroView SCADA system. Although the MVX Object Server allows for other communication protocols to be used, only a small number of additions had to be made to allow the Metaserver Protocol to transmit the extra data now being defined on the server (mainly the concept of ‘objects’ and the relationships between them). It was therefore decided that another part of this project would be to make the necessary client/server changes to allow for the transmission of object graphs via the Metaserver protocol. Again this was another decision where it was either do more work now that can be leveraged in the long run, or do a quick throw-away implementation that will require repeat work later.

2.3 Client Visualization Software

The MVX product currently under development has heavily leveraged a third party 3D engine and Integrated Development Environment (IDE) known as Unity3D. It provides a modern, high performance 3D engine, and exposes a Mono (open source .NET runtime) scripting engine. The Mono scripting engine was the selling point for MVX, as it allowed for re-use of the existing .NET MacroView client libraries that provide communication to the server via the Metaserver Protocol. Therefore Unity3D was at the top of the short list of 3rd party client visualization tools for this project. Although this project would not utilize the traditional full scene rendering of a 3D engine, a 3D engine is still very useful for simplifying the augmented reality process – when the live camera feed and 3D engine ‘camera viewpoint’ settings are synchronized correctly, the 3D scene’s positions become synchronized with the real world coordinates, allowing easy placement of augmented reality data visualizations on top of the live camera feed backdrop.

The only drawback is the lack of video support within Unity3D. While it does support playing videos in Ogg Theora format, its whole API is structured around a fixed length video format (no streaming support). While this limitation did mean an additional step was required (a video feed transcoding service to convert the live camera feeds into short Ogg Theora files dynamically), the benefits of the re-use of the MacroView client communication code, and the ease of use of implementing augmented reality within a 3D engine meant Unity3D was still the optimum choice.

To reduce complexity for this prototype, it was designed from the start to utilize fixed location cameras of known orientation. This reduces the AR implementation to camera/3D scene synchronization, without requiring computer vision methods of feature detection or Simultaneous Localization and Mapping (SLAM) techniques.

3. Project Implementations

3.1 MVX Object Server

The MVX Object Server implementation leveraged an in-house .NET domain model framework library known as the Torq Framework to reduce the amount of ‘plumbing’ code required for this project. The Torq Framework abstracts the data source of objects and their properties from the class itself, via an ITorqObjectSource interface, and was an ideal basis for the MVX object server where an object graph needed to be defined and persisted for clients to consume – it allowed for existing persistent storage implementations (e.g. An Object Oriented Database called Matisse) to be plugged in without requiring any new code to be written. However, to pull SCADA system data from the multitude of existing systems, and update the
object graph exposed on the MVX object server, additions were required at the Torq Framework level. The concept of an *Externalizable* base class was defined, which abstracts the concept of external data sources that can update individual properties of an object. Individual implementations of these *IExternalDataSources* can then be created and injected as needed. For this prototype system, external data sources were implemented for retrieving data from a MacroView SCADA system, from an XProtect Video server, and from an Excel Workbook.

### 3.2 Metaserver Protocol Additions

Once the persistent domain model with external data sources component was complete, the next phase of the server implementation was to expose this data structure to clients via the Metaserver Protocol, and to provide real-time updates to clients as changes to the object graph occurred. Although the serialization and communication libraries already existed for other Metaserver interactions, additions had to be made to support the concept of replicating an object graph across the network. A range of new message classes were defined to transmit object definitions and instances, as well as the discrete property values and relationships between these objects. With both the server and client side using the same .NET communication libraries, the additional message classes were only written once, and shared by both the server and client libraries.

![Figure 2](image.png)  
*Figure 2* A network model of the communications between the external datasources, the MVX Object Server, and the MVX clients. Note that this AR project only includes the ‘MVX AR Client’ within the MVX Clients group. The other types of clients are included to illustrate the multipurpose nature of the MVX Object Server.
3.3 Augmented Reality Client Software

The first task of getting video feeds into Unity3D was simplified by using jpeg image streams from the XProtect Video server. This reduced the complexities of dynamically transcoding and creating small Ogg Theora video files for the first phase of development, however it did increase bandwidth costs significantly, especially if requesting a high frame rate.

The next phase of development consisted of writing the Unity3D/Mono code that utilized the MacroView .NET communication libraries to receive the MVX Object Server data and setup the AR scene as required. This included physical camera/3D scene camera synchronization, and virtual object creation. The virtual object creation phase consisted of creating the various virtual graphics that represent the available data, and overlaying these graphics in their respective game coordinate locations so that they match up with the real world video feed.

This produced a basic augmented reality system, but no virtual object occlusion culling occurred. For this prototype, it was decided that the system would have to be provided with basic model data of the physical environment the camera was viewing. The augmented reality system then places these environment models into their respective game coordinate positions, and using the built in 3D engine occlusion culling, only displays the augmented reality graphics if they are not blocked by these environment models.

4. Conclusions and Future Work

The augmented reality industrial control visualization system performed all tasks as set out in the initial design specification to an acceptable prototype standard. The AR visualization component did not receive as much time as had been planned for, however the MVX Object Server is now a high quality, highly flexible and extensible server system.

Possible future work for the industrial control AR visualization system would mainly focus on the visualization client. A commercial level implementation of the visualization client (compared to the current prototype implementation) would be required before the product could be sold, and would most probably require integration with the other MVX products. This option has in fact already been discussed, and a brief design has been drawn up where MVX abstracts the concepts of virtual views, live camera views, and augmented reality views, becoming a series of operator controlled ‘views’ that can be interchanged as required.

A longer term goal for industrial control augmented reality includes the computer vision aspects of SLAM to allow for on-site operators to view site data via a hand-held tablet PC with camera held up towards the equipment of interest – the screen would display the live video feed coming from the PC’s camera overlaid with the equipments related SCADA system data.

6. References
