Scan Conversion Software Architectures

A summary of scan conversion architectures supported by the SPx Development software

Summary

The SPx library provides a number of methods of adding scan converted radar video into a graphics application under Linux or Windows.

This application note summarises the methods and presents the key features and benefits of each approach.

SPx supports a number of methods of adding radar scan conversion into a graphic application (for Linux or Windows).

The different methods are designed to accommodate different application requirements or constraints imposed by the existing software or architecture.

This note summarises the methods available and provides guidelines to choose the best solution.

The starting point for the different solutions is the assumption that there exists, or will be developed, a graphical application under Linux or Windows that is required to display scan converted radar video.

The application will likely display maps, targets and support normal user-interface controls. There will be one or more windows in the application where it is desired to *insert* radar video.

Once inserted, the client will in general need to control the scale and view of the radar image to match the scale and view of the graphics application. The following methods for the addition of scan conversion are supported:

Method 1: Local Scan Conversion with the SPx Class Library

In this method, the client application uses the SPx C++ class library to add scan conversion capabilities.

The application creates SPx objects within the application to handle the radar input, optional processing, and scan conversion.

The scan conversion process outputs its radar image into one or more windows managed by the client application.

The client calls functions in the class library to manage the display of the radar video (window size, view geometry, fading, colour etc).

This method of adding scan conversion is the standard approach and provides greatest flexibility, since the application has full access to all objects and functions in the class library.

Method 2: Local Scan Conversion with RDC

In this method, the client application uses the services of the Radar Display Coprocess (RDC), which runs as a self-contained Linux or Windows process.

The RDC is supplied by Cambridge Pixel as a ready-built application, which is run as a background process.

The client application connects to the RDC using a lightweight C++ class interface for messaging.

The RDC is able to insert radar video into windows created and managed by the client. When the client wants to change the parameters of the scan conversion process, it simply sends a message to named objects in the RDC using a simple messaging protocol.

When used with Windows, this approach is supported by the Microsoft .NET interface.

Method 3: Remote Scan Conversion with the Scan Conversion Server

In this method, the scan converter runs on a remote network server and delivers regular updates of scan converted data to the client.

These regular updates allow the client to build a radar image that shows continuously updating radar sweep, such that the quality of the radar image is for all practical purposes no different to that obtained from Methods 1 and 2.

A scan conversion server is associated with a single client, and that client controls the remote scan conversion process through a set of API calls.

This approach offers the same range of features and control of the radar image as Method 1. The main difference is that the scan converter is running in a server and the scan converted radar image is distributed back to the client over a network.

Method 4: Remote Scan Conversion with the Radar Image Server (RIS)

In this method, the scan conversion runs in a remote server (like method 3), but this time a single server delivers a fixed radar image for multiple clients.

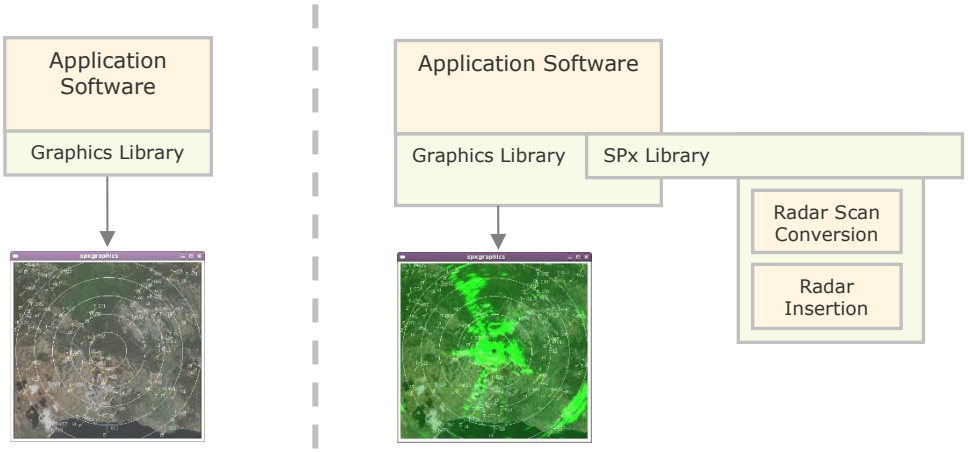
The client is unable to control the view delivered by the server and simply retrieves the view that the server is programmed to deliver.

This provides less flexibility in the client, but it also simplifies the client interface. The RIS can deliver the radar image into a standard internet browser or else into custom developed application.

Which method to Choose?

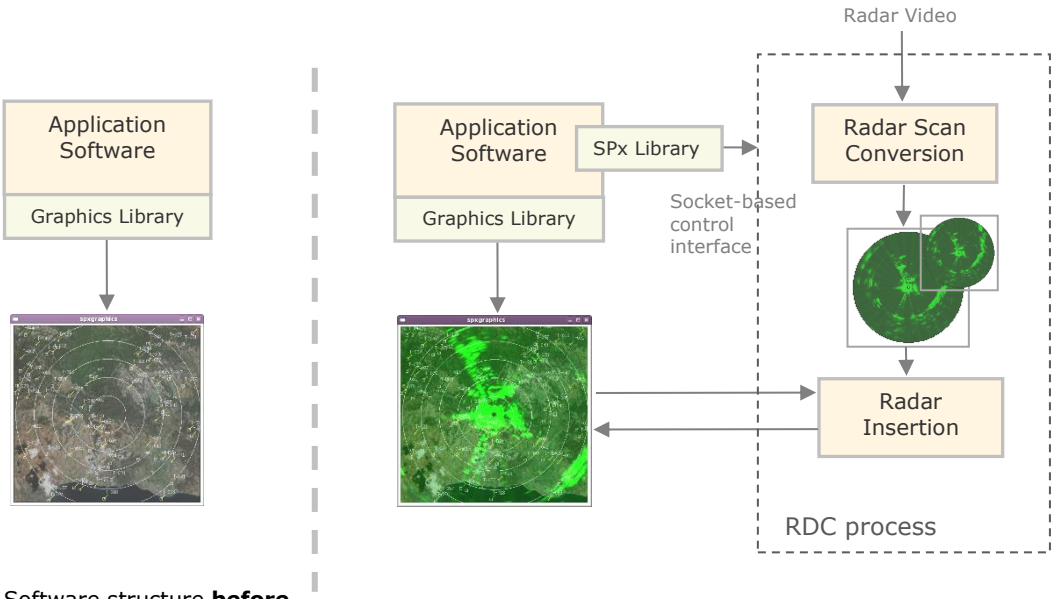
Method 1 provides the greatest flexibility, since the full class library is available for the client application. The scan conversion is linked as part of the client application and it will run in its own thread, which on a modern processor will typically mean that can run in its own core. The client application will need to be recompiled and relinked if there are any changes to the SPx library. It might sometimes be preferable to separate the application and the scan converter into different processes - this is Method 2.

The benefit of Method 2 over Method 1 is that the SPx processing is not part of the client application. A new enhanced RDC could be used without even requiring a relink of the application. Further, the client processing and the RDC scan converter are cleanly separated into different processes, enhancing the robustness of the solution. The diagrams below show the software components in methods 1 and 2.



Software structure **before** adding SPx scan conversion

Software structure **after** adding scan conversion using SPx C++ class library (Method 1)



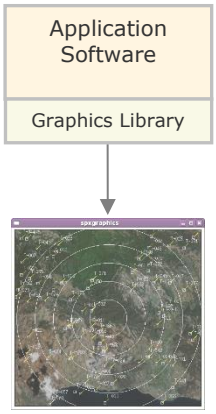
Software structure **before** adding SPx scan conversion

Software structure **after** adding scan conversion using RDC (Method 2)

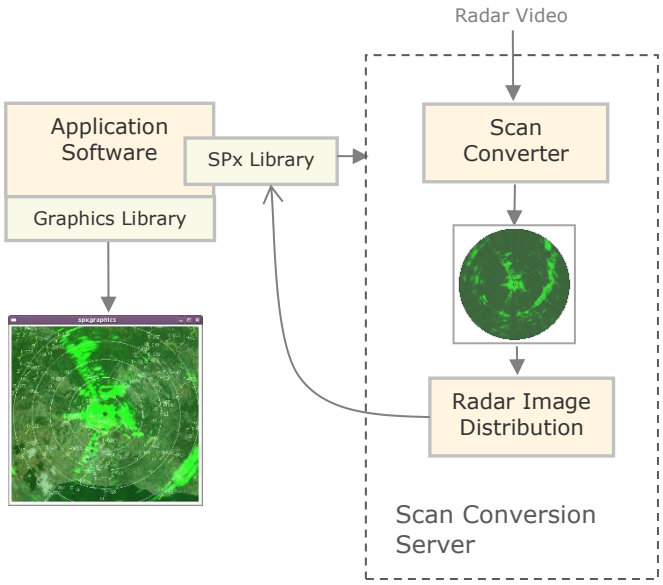
Method 3 uses a remote scan converter. This means that the polar radar data never reaches the client, which only sees the scan converted image data. If the polar radar data has a significantly higher bandwidth than the scan converted radar image, then this approach can reduce the volume of network traffic arriving at the console. In method 3, each scan converter is associated with a single client, which provides the control. Requests from the client to change the radar view, for example, are sent as network messages to the scan conversion server, which then scan converts and distributes the new view.

Although Method 4 appears similar to Method 3, there are some important differences. Method 4 uses the Radar Image Server (RIS), which is a standard server application provided by Cambridge Pixel. Significantly, the client has no control over the scan conversion server. The server sends out one or more channels, each comprising a fixed view of scan converted data. This image data is typically delivered 4 times per radar sweep.

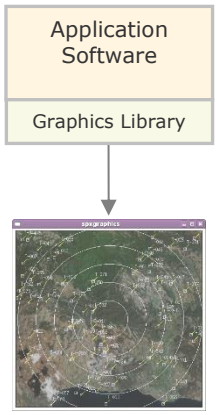
The server can output 1, 2 or 4 channels (license options) and any number of clients can view the data either through a standard internet browser or using a custom application using Cambridge Pixel supplied library routines to read the data.



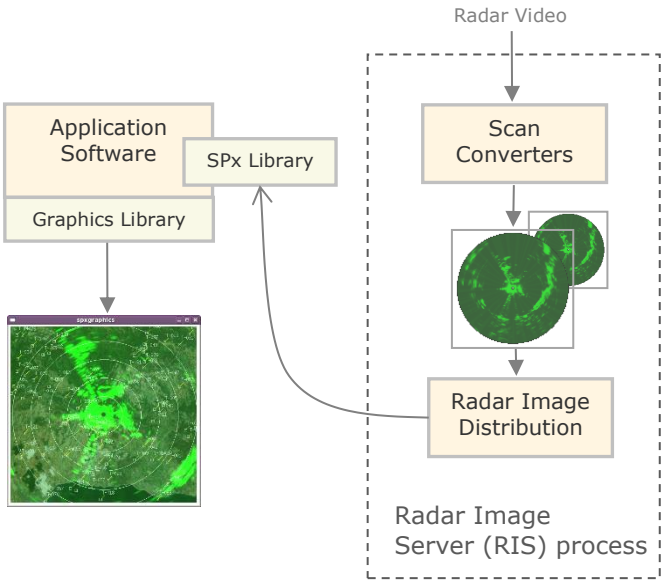
Software structure **before** adding SPx scan conversion



Software structure **after** adding scan conversion using Remote Scan Conversion Server (Method 3)



Software structure **before** adding SPx scan conversion



Software structure **after** adding scan conversion using Radar Image Server (Method 4)

Licensing Requirements

SPx Scan Conversion requires runtime licenses, but the licensing issues with the 4 methods are different and summarised by the following table:

	Needs SPx Development License?	Runtime license?
Method 1	Yes	Yes, per client application
Method 2	Yes	Yes, per client application
Method 3	Yes	Yes, per scan conversion server (which can support one client display).
Method 4	No	Yes, per Radar Image Server (1, 2 or 4 channels). Client licenses not needed.

< End of document >