User Manual for Kakadu's "kdu_winshow" and "kdu_macshow" Applications— last updated for version v7.0

David Taubman, UNSW

1 Introduction

This document is designed to accompany the two demonstration viewers which come with Kakadu, namely "kdu_winshow" and "kdu_macshow". The former is a JPEG2000 file viewer for Windows platforms, running anything from Windows XP on, while the second is for MAC platforms running anything from MAC-OSX 10.5 on, with Intel or PPC processors. All source code to each of these tools is provided with licensed copies of Kakadu and may be used to aid in the construction of more elaborate applications. However, both viewers already possess a very full list of features. At the time of this writing, both viewers offer exactly the same functionality, the same set of menu options, and similar accelerator keys, differing primarily in their Graphical User Interface code, which is inherently operating-system dependent. They are both implemented on top of powerful platform-independent Kakadu API's, such as '*kdu_region_compositor*', '*kdu_region_animator*', '*jpx_source*', '*mj2_source*', '*kdu_client*' and the like.

We use the terms "*kdu_macshow*" and "*kdu_winshow*" to distinguish between the MAC and Windows versions of the viewer in this documentation. However, on the term "*kdu_show*" is used where such distinction is not important (most of the time). The following is a partial list of features supported by these viewers:

- Open any JP2, JPX, MJ2 or raw JPEG2000 codestream file.
- Remotely browse any JP2, JPX or raw JPEG2000 codestream file via JPIP
  - use "*kdu_server*" to serve content very efficiently for interactive browsing
  - supports HTTP transport, HTTP-TCP transport, HTTP-UDP transported sessions and sessionless HTTP browsing.
- Pan, zoom and re-orient the view using a variety of controls, including fine zoom factors (not just powers of 2).

- Multi-threaded decompression to exploit available CPU resources.
- Very low memory consumption – assuming the source file contains random access information, even Terapixel images can be interactively viewed using only a few MBytes of system memory.
- Walk through or play arbitrary JPX animations and MJ2 video tracks
  - Full control over playback is offered via menu commands, accelerators and a familiar control bar.
  - Multi-image sources such as JPX files that represent 3D volumes can be treated as animations even if they do not provide frame composition and animation instructions.
  - JPX animations can be browsed remotely via JPIP; however, this feature does not currently involve requesting content ahead of time so real-time playing of JPIP served content should be done at very low frame rates and you should go back and forward over the content to see new data as it arrives.
  - As of version 7.0, "*kdu_winshow*" uses DirectX during animation and video playback to avoid glitches and tearing. The "*kdu_macshow*" viewer generally performs well, but relies on the native double buffering provided by Acqua to avoid tearing.
- Extensive support for displaying and editing metadata
  - Both viewers present a metadata catalog sidebar when appropriate (can be opened and closed manually if you like)
  - Both viewers provide metadata overlays for the imagery so that you can visualize regions of interest as and when they become available, with user control over the brightness and dynamics of the overlay content
  - Both viewers provide a detailed metadata editor for individual items, which can also launch external editors for content like XML.
  - Both viewers provide a graphical region-of-interest editor with three editing modes, path sketching, scribble and region filling capabilities.
  - Both viewers provide extensive support for discovery and interpretation of cross-references (links) within the metadata structure of a JPX file.
  - Both viewers use JPIP in a sophisticated way to request metadata of interest on-demand during interactive browsing.
- Metadata-driven animations

  - Both viewers can dynamically instantiate and play animations based on metadata selected from the catalog sidebar. These animations automatically walk the viewport through regions of interest, animation frames, etc., based on the selected metadata. You can still interactively zoom, rotate and resize the window while all this is going on.
  - This all works during remote browsing with JPIP, although as mentioned above you are recommended to use low frame rates and select the repeat option so that you can see the content improving in quality over time as data arrives.

- Both viewers support saving of content to JP2 and JPX files, regardless of how the content was obtained and regardless of how complete the content might be if recovered via interactive JPIP browsing.
- Both viewers support multiple windows
  - You can open a new window into a local or remotely browsed image based on a selected region of interest.
  - You can open a new image (local or JPIP served) by clicking on hyperlinks in the metadata catalog sidebar.
  - When working with JPIP, all windows associated with a given source image/animation share a common cache so image content improves in all windows as more data arrives, even though separate JPIP request channels are used for each window's viewport/ROI.
- Both viewers support low-level analysis of imagery, codestream properties and metadata/file structure
  - You can walk through the individual image components in a codestream
  - You can inspect all coding parameters of any codestream in the source, getting help on each type of parameter attribute simply by clicking on it.
  - You can inspect the metadata structure of the file in a "*metashow*" window that is cross-linked to the metadata editor and the image view where appropriate.
  - The status bar at the bottom of the image view window provides significant additional information, such as dimensions, memory consumption, JPIP cache contents, frame numbers and so forth – you can switch between different status views using menu/accelerator options.

The above list of features is not exhaustive. Before plunging into a more detailed discussion of the "kdu_show" viewers, it is worth re-iterating that practically all of the above features are implemented using platform independent high level objects that ship with licensed copies of the Kakadu software, using as little platform-dependent GUI glue as possible. For example, even the graphical region-of-interest editor for JPX metadata is built on top of a platform independent editing object that is part of Kakadu's JPX metadata API.

2 Basic Operations

2.1 Opening and Closing Files

The File menu may be used to open and close files. Alternatively, you may find the "ctrl-o"/"cmd-o" accelerator keys useful for opening files quickly. Opening a new file automatically closes any existing open file; however, you can first open a new window via the menu or via "ctrl-n"/"cmd-n".

From a Windows command prompt, you can supply the name of a file directly as in

```
kdu_show file-to-open
```

on the command-line. You can do something similar on the MAC using a command-line of the form

```
open -a kdu_show file-to-open
```

or you can use the simpler Windows syntax if you first define the following alias in your .login script:

```
alias kdu_show="open -a kdu_show"
```

The "*kdu_show*" tool can open JP2 files, JPX files, unwrapped JPEG2000 codestreams and Motion JPEG2000 (MJ2) video files.

2.2 Panning, Zoom and Re-orienting

The "*kdu_show*" tool started out as a sophisticated demonstration application to showcase some of Kakadu's capabilities for incremental decompression of large images, based on spatial regions of interest (view ports), as well as Kakadu's ability to decompress images in a geometrically transformed domain. In fact, Kakadu's codestream management architecture has always allowed compressed JPEG2000 images to be accessed as though they had been compressed as smaller image, rotated and/or cropped versions of themselves. As a result, the core "*kdu_show*" features have always been those of panning, zooming, rotating and flipping a viewport into the compressed image.

Panning may be accomplished via the scroll-bars, or using the arrow keys. The PageUP and PageDown keys may also be used for panning (Windows only). The image may also be dragged around with the mouse while the left mouse button is depressed and the shift key held down. The mouse wheel provides another means of panning around.

Zooming may be accomplished via the View menu or with the "z" (zoom in) and "ctrl-z" (zoom out) accelerators. By default, zooming is centred about the middle of the current viewport, but you may define an alternate focus box using the method described below. Zooming out is accomplished using the multi-resolution attributes of the discrete wavelet transform (DWT). However, you can zoom out further than the lowest available DWT resolution and you can zoom in further than the highest DWT resolution, with additional zooming being handled during the rendering stage, as opposed to decompression itself – this is rarely required and is inherently slower.

Rotation and flipping may be accomplished via the View menu or with the "[", "]", "_" and "|" accelerators. Kakadu accomplishes these effects by decompressing the imagery in a reverse and/or transposed order, as a highly efficient core capability, rather than just decompressing the image and then subsequently reorienting it.

The application automatically resizes its window to match the image size, unless the image is too large to be viewed in the current window, in which case a restricted viewport is displayed. You may adjust the window size via the usual resize handles. Alternatively, and often more conveniently, you may enlarge or shrink the view window quickly using the "w" and "s" accelerator keys.

Special note on zooming: JPX files can contain compositions of codestream imagery, where each composed piece has arbitrary scale factors. The "*kdu_show*" tool handles such imagery correctly, regardless of the implied scaling factors, but there might be no one global scaling factor which results in optimal rendering of the entire composition. Also, the policy in "*kdu_show*" is to keep the global scaling factor constant as you navigate to different compositing layers and/or composited frames. As a result, you may find that the scaling factor applied to a particular codestream image component of interest is not ideal. Ideal scaling factors are integer expansion factors and power of 2 decimation factors. Whenever you use the "z" (zoom in) and "ctrl-z" (zoom out) function, the zoom factor is automatically adjusted to one which is ideal for the top-most or dominant imagery found within the view window or focus box (see below). However, you may also find the "alt-z" (optimize zoom) function very useful. It makes the smallest possible change to the current global scaling factor so as to optimally render the content which is dominant within the view window or the focus box (see below).

In practice, you are unlikely to notice the difference between ideal and non-ideal scaling factors visually because the underlying API's automatically decompress content with non-ideal scaling factors at a higher resolution and then render down to the required scaling factor on the fly, using high quality anti-aliasing filters. However, whenever "*kdu_show*" needs to do this, the rendering speed will be up to four times slower which might be noticeable on some machines. Unlike many image viewers that you may be familiar with, "*kdu_show*" does not cache any rendered or decompressed content; it generates everything on the fly in an incremental fashion, often using very little system memory at all.

Special note on zooming vs. display scaling: The "*kdu_show*" tool also provides you with a special "*Scale X2*" feature, which may be accessed via the View menu. While this may seem very similar to zooming, there is an important distinction. The "*Scale X2*" feature causes each rendered image pixel to be displayed using a 2x2 block of display pixels, regardless of the current zoom setting. For example, zooming out from the image (so that it is being rendered at 50% resolution) causes the highest frequency subbands and the last stage of DWT synthesis to be dropped from the compressed imagery. With the "*Scale X2*" you can visualize the resulting imagery more clearly, especially on displays with very small pixels.

2.3 Focus Boxes

Focus boxes have at least 4 uses in "*kdu_show*": 1) they define a centre for zooming; 2) they define the region of interest during remote image browsing with JPIP (see Section 4); 3) they highlight spatial regions of interest defined in the metadata; and 4) they provide a starting point for defining new spatial regions of interest to be added to the metadata using the methods described in Section 5. As of version 7.0, focus boxes display slightly differently depending on whether they are manually defined are derived from some existing metadata. For example, if you double click on an item in the metadata catalog sidebar that happens to have region-of-interest associations, the image view will be navigated to the relevant content and a focus box will appear around the relevant region. However, if you subsequently move or redefine the focus box, it will change in appearance.

To define a focus box, hold the left mouse button down while dragging the mouse to define the region of interest.

To remove a focus box, you may: 1) use the Focus menu; or 2) hit the "f" accelerator key.

By default, the focus box will be displayed using both a highlighting technique and a dashed outline. The highlighting technique makes the rest of the image slightly darker, and the focus box slightly lighter. You can turn off the highlighting by using the Focus menu, or hitting the "h" accelerator key.

You can widen or shrink an existing focus box using the Focus menu, or the "shift-w" and "shift-s" accelerators.

You can pan the focus box using the arrow keys, while holding down the shift key. Basically, the accelerators which affect the focus box are identical to those used to manipulate the view window, except that the shift key is held down.

2.4 Quality Layers

JPEG2000 codestreams may contain multiple embedded image qualities, which form the foundation of quality progressive remote image browsing. By default, local files are opened at maximum quality, but it can be useful to see the visual quality associated with each compressed quality layer. To reduce the number of quality layers you may use the Quality menu or the "<" accelerator. You may increase the quality again using the Quality menu using the ">" accelerator. These manipulations automatically change the status bar to one which displays the number of quality layers which are in use.

When playing video content, you might find that displaying all of the available quality layers is too slow. Reducing the quality typically increases rendering speed dramatically, especially when the original content was losslessly compressed. Of course, this is only possible if multiple quality layers are available in the source content.

2.5 Navigating Between Image Elements

The "kdu_show" tool has 3 fundamental modes of image display, as follows:

Single Component Mode: In this mode, only a single image component is displayed as a greyscale image. Image components typically correspond to colour channels (e.g., red, green, blue, or luminance, and chrominance channels), but they may correspond to alpha blending channels, palette indices, or anything else.

To enter single component mode, use the View menu or hit either of the "1" or "+" accelerator keys. While in this mode, you may navigate forward and backward within the image components of a given codestream by using the View menu or the "+" and "-" accelerators.

While in single component mode, you may navigate amongst the codestreams in the file (JPX and MJ2 files may have any number of codestreams) using the RETURN and BACKSPACE keys. Alternatively, you may use the View menu or the animation control bar that appears as needed at the bottom of the image view window.

Single Compositing Layer Mode: In this mode, only one compositing layer is displayed, usually as a colour image, or alpha blended against a solid white background. A JP2 file contains exactly one compositing layer, which is the rendered colour image. A JPX file may contain any number of compositing layers. A raw codestream is treated as having exactly one compositing layer, corresponding to our best guess regarding the intended display image.

A Motion JPEG2000 (MJ2) file may contain one or more video tracks, each of which is treated as a compositing layer. This is sensible since MJ2 tracks can be composed onto a richer surface in the same way that JPX compositing layers can be composed. The fundamental difference between MJ2 video tracks as compositing layers and JPX compositing layers is that the former typically involves multiple images (video frames), all with the same dimensions, whereas the latter involves only a single image.

To enter single compositing layer mode, use the View menu or hit the "L" accelerator key. While in this mode, you may navigate forward and backward amongst JPX compositing layers or amongst frames of a current

MJ2 video track, by using the RETURN and BACKSPACE keys, the View menu or the animation control bar. To move to a different MJ2 video track, use the View menu.

Composited Image Mode: In this mode, the application displays what it believes to be the intended image. For JP2 files and unwrapped raw code-streams, this is identical to the result displayed in single compositing layer mode. If a JPX file contains a composition box, however, the composition instructions are used to build the composited image which was intended by the content creator. For MJ2 files, this mode is currently equivalent to single compositing layer mode, so you can only view one video track at a time. However, this is a limitation only of the "*kdu_show*" demo applications; the underlying '*kdu_region_compositor*' object, which does all rendering work, supports composed video tracks.

When a new file is opened, "*kdu_show*" starts out in the composited image mode, but if you have changed the mode to single component or single compositing layer mode, you may get back to composited image mode by using the View menu or hitting the "c" accelerator.

While in composited image mode, you may navigate forward and backward between composited frames (animations contain multiple frames, each of which is a composited image) by using the RETURN and BACKSPACE keys, the View menu, or the animation control bar.

2.6 Status Information

Some summary information is displayed at the bottom of the application window on its status bar. The status bar is divided into fields. The information displayed in these fields also depends upon the status mode, which may be toggled using the Status menu, or via the "ctrl-t"/"cmd-t" accelerator key. In particular, you can currently toggle between the following status modes.

3 Multiple Windows and Command Broadcasting

The "kdu_show" application often displays only a single window, which may consist of multiple panes: image view; status bar; JPIP progress bar; animation bar; metadata catalog side bar; and metadata pastebar. When this window is closed, the application itself closes. More generally, though, "kdu_show" can manage any number of windows, each of which might correspond to the same source file or perhaps different ones. The application automatically closes only once all windows have been closed.

Multiple windows serve many purposes. One useful feature of "kdu_show" is that you can open a new window around a selected focus region. This can be done using the "ctrl-d"/"cmd-d" accelerator or via the Window menu. The new *duplicate* window inherits all the viewing state from the window that was the source of the duplication operation. Duplicate windows are of some interest for local files, but of huge value when browsing content remotely via JPIP. In this latter case, the local cache of JPIP served data is shared by all windows that are associated with the same source. Moreover, each window corresponds to a distinct physical or virtual channel to a single session on the remote server (assuming that the "NONE" transport protocol was not selected). This ensures that communication is very efficient and that the requests generated on behalf of each window work together to enhance the overall browsing experience for all windows, in the most efficient manner possible.

You can use the "ctrl-n"/"cmd-n" accelerator of the Window menu to open a new empty window at any time, from which you can open imagery in all the usual ways. Multiple windows are opened simultaneously if you pass a list of source files to "kdu_show" when launching it, or if you select multiple source files during a File->Open operation.

On the MAC, it is pretty much impossible to get multiple instances of the "kdu_macshow" application to run concurrently. Instead, each new source that is opened automatically appears as a new window within the application. This is, of course, the recommended behaviour for all MAC applications. There are quite a few advantages to this. One of these is that the application is able to warn you of unintended consequences if you are about to save data from one window over a file that is being used (perhaps as an externally linked resource) by another window. The other, of course, is the ability to consistently share JPIP cache information and server sessions for remotely browsed content.

On Windows, you can either open a new copy of the "kdu_winshow" application or you can open a new window within an existing instance of the application. This can be confusing, since there will not be any visual difference between the two cases (except perhaps within the task bar). For the reasons mentioned above, it is generally preferable to open multiple images as separate windows within a single instance of the application.

One final advantage of multiple windows that is worth mentioning here is that you can arrange to "*broadcast*" menu commands that are directed to one window to the other windows; this includes menu commands that are induced by accelerator keys. The Window menu provides you with two options for doing this: broadcast the next command only; or broadcast continuously until the feature is disabled. As an example, with this feature enabled, each time you zoom or rotate one window's view, all the other windows will do the same. An even more useful example occurs in the playing of videos or animated content. In this case, broadcasting the "play forward" menu command to all windows will cause them all to start playing at precisely the same instant – similar benefits arise for other animation control commands.

4 Remote Image Browsing with JPIP

By and large, images may be accessed remotely from a JPIP server (such as that implemented by the "*kdu_server*" utility) exactly as if they were local files. Rather than experiencing significant delays while the image downloads, you will instead observe incrementally improving quality as image data is incrementally transferred by the server. JPIP servers focus the transmitted data around the region defined by the focus window so the quality in that region increases more rapidly. If no focus window is defined, it is taken to be the current view window. Once sufficient data has been received to reconstruct the image within the focus window without loss, at the current display resolution, the server enters the idle mode and will not transmit any further data until the image region, resolution (zoom factor), codestream, image components, compositing layer or animation frame are changed.

It is important to realize that the viewing tool is only loosely (asynchronously) coupled to the client component and, ultimately, to the server utility. You are always free to pan, zoom and otherwise navigate the display to any part of the image, regardless of whether any data has been received from the server to render that region. After some time, the server will adjust its transmission pattern to serve the new image region/resolution which is currently of interest. Of course, if the server has disconnected, no amount of navigation will bring in new data, so you will generally be left with non-uniform image quality which reflects the amount of time spent browsing in different image regions.

It is also important to realize that the JPIP server is allowed to shrink the region of the image for which it will serve data, if the region is too large to be served in a quality progressive fashion without consuming excessive server memory resources. If this happens, the changed region will be reflected on the view window through a new focus box. You may move this modified focus box around, to change the region to which the server's transmission is customized, but you may not enlarge the focus box without having the server force it back down again to its size limit.

In the current version of "kdu_show" JPIP service preferences are sent to the server in a sophisticated manner to make browsing as natural as possible. If a focus window of interest has been defined, the service preference is to allow the server to reduce the focus window if that turns out to be necessary to provide a smoothly improving quality progression over the window – up to the server. If there is no focus window, the server is asked to deliver the entire viewport, regardless of whether this can be done in an ideal fashion or not, since the user does not appear to be interested in interactively selecting content of interest. If there is a user defined focus box that extends over multiple actual images (in a JPX composition), the server is again asked to deliver everything in the focus box, regardless of whether this can be done in an ideal quality progressive fashion or not. Kakadu's server respects these preferences and still manages to do a good job in all cases. Other servers, or older versions of Kakadu's server might behave differently, so that a reduced focus box might appear even when one was not defined by the user, to highlight the region that is actually the subject of content streaming.

4.1 Regular Image Browsing

To open a remote image, use the File->OpenURL menu item, or start "*kdu_show*" with an appropriate URL on the command line (only for Windows). The "kdu_show" application should also open automatically if an appropriate hyperlink is selected within Internet Explorer (Windows) or Safari (MAC) – see below. The File->OpenURL dialog box provides several fields for you to fill out. For regular image browsing, enter the name of the file you want to access in the "*Object requested from server*" field and enter the name or IP address of the server machine in the "*Server name or address*" field. You may append an optional colon-separated port number to the server name or address, if your server is not listening on the default port 80. IPv6 addresses are fully supported if you are familiar with the standard syntax.

One of four closely related JPIP transfer protocols may be selected via a drop-down list. For the most efficient communication, the "*http-tcp*" option is usually recommended – "*http-udp*" might be more efficient, but is currently experimental and most server's are unlikely to support it or have it enabled by default. The "*http-tcp*" transport option requires the establishment of both an HTTP and a regular TCP channel, which might not be permitted by some institutional firewalls. In this case, select "*http*" as the next best option. The "*none*" option is primarily for experimental purposes. It is much less efficient, especially for the server. Note; if you select an advanced option that is not supported by the server, the next less advanced option that is supported should end up getting used automatically. For example, "*http-udp*" falls back to "*http-tcp*", then "*http*" and ultimately "*none*" if the server handles the requests correctly.

If your organization requires all external traffic to flow through an HTTP proxy, select the "*http*" option and put the name or address of your HTTP proxy server in the "*Proxy name*" field – you may need to consult your system administrator to find out the name of your HTTP proxy. Proxies reduce the communication efficiency and responsiveness, so use them only if necessary.

You may elect to have all your image browsing results saved in a local cache directory, by entering the name of this directory into the "*Cache dir*" field. The advantage of this is that when you open the same image again on the same server, or a compatible server (one which assigns the same unique target identifier to the image resource), the previous browsing results will be reused and that data will not be sent over again by the server.

An alternate way to open an image on a remote JPIP server is to supply the URL on the command line (Windows only). The URL must commence with either "*http://*" or "*jpip://*" as the protocol identifier prefix, followed by the server host name (or IP address), a single "/" character, and then the resource string. The syntax for the resource string and the server name are identical to those used by the File->OpenURL dialog box. The alternate "*jpip://*" protocol identifier is provided so that the "*kdu_show*" utility can be fired up automatically from a web browser whenever a URL with this protocol identifier is encountered. In fact "*kdu_winshow*" registers itself with the Windows Internet Explorer as the target for such URLs, the first time it is ever used. Similarly, "*kdu_macshow*" registers itself with the Safari web browser. You should be aware that when URLs are supplied on the command line or opened via a web browser, "*kdu_show*" obtains the remaining information (proxy server, cache directory, protocol variant), based on the last values supplied via the File->OpenURL dialog. For this reason, it is advisable to use this dialog to configure the application first, before opening URL's from the command line or by clicking on JPIP URLs on a web page.

You may explicitly disconnect from a JPIP server, without actually closing the image, by using the File->Disconnect menu item. In some cases, this may happen automatically if a server decides that your connection has been inactive for some time.

4.2 One-Shot Image Browsing

To implement the functionality of an image browser, "*kdu_show*" uses Kakadu's '*kdu_client*' object to generate a sequence of much more complex HTTP requests than that supplied via a typical command line URL. These much richer requests are generated using either HTTP POST or HTTP GET with query parameters appended to the URL, separated by the usual "?" query separator.

As an alternative to interactive image browsing, you may supply one of these more complicated URLs directly by including the "?" query separator and the query fields of interest. In this case, only one request will be delivered to the server and no amount of interactive navigation within the image will cause any further requests to be sent to the server. The server will continue to transmit data relevant to the original request until all relevant information has been sent or the server times out or is disconnected. As a result, interactive navigation and image data delivery are entirely disconnected processes; you may be receiving data relevant to a small image region at high resolution, but be viewing the image at low resolution (zoom factor) or vice-versa. You may even be viewing a completely different codestream within the data source and so not see the effects of newly arrived image data.

The main purpose of this mode of behaviour is for server debugging and for modeling the behaviour of JPIP when all imagery is generated in response to a static URL, which might be embedded in an HTML page.

You may also enter single-shot URLs via the File->OpenURL dialog. To do so, simply append the "?" query separator and relevant query parameters to the string entered in the "*Object requested from server*" field of the dialog box. Notice that the File->OpenURL dialog contains a check box that lets you specify whether interactive browsing within a source that was opened using a single-shot URL should generate additional requests to the server or not.

To use single shot URLs you will need to be familiar with the JPIP syntax. It is not the purpose of this document to describe this syntax, but the JPIP committee draft may be downloaded from "http://www.jpeg.org/CDs15444.html." You may also examine the JPIP requests delivered to the Kakadu server during normal interactive image browsing by supplying the '`-record`' command line option to "*kdu_server*".

5 Video and Animations

The "kdu_show" application automatically instantiates an animation control bar below the image view panel (actually below the status bar), whenever the content suggests that it can be played as an animation. As explained in Section 2.5, there are three fundamental modes of image display. Video/animation playback is not available in *single component mode.* Most of the time, you will be in *composited image mode*, where the image view shows JPX composited frames (if they exist) or MJ2 video frames. In this case, video/animation playback is available if there are multiple such frames in the source. The *single compositing layer mode* is currently identical to composited image mode for MJ2 video. For JPX files, single compositing layer mode shows individual JPX compositing layers, as opposed to composited frames. In this case, animation playback is available so long as more than one compositing layers exist in the source. Animation playback of individual compositing layers has no timing information, so the animation bar's progress slider reveals compositing layer indices (starting from 0) instead.

Animations can be controlled via the animation bar or via the Play menu – the latter is of interest primarily in that it offers accelerator keys for actions such as play-forward, play-backwards, stop-playback and changing the playback rate. All of these options may be controlled, however, by clicking buttons or dragging the timescale slider in the animation bar.

From version 7.0, "kdu_show" does allow you to play back JPX animations that are being accessed remotely via JPIP. However, the full functionality of an efficient remote video player is not yet quite implemented. Specifically, the implementation does not currently request video/animation frames from the server ahead of the point at which they are required by the playback machinery. With conventional, non-scalable compressed media, this would hold up the playback machinery until frame data arrives from the server, but with JPEG2000 and Kakadu, the ability to decompress and render content of interest is almost totally independent of the arrival of content from the server. As a result, the networking and rendering components work independently and you will find that unless the frame rate is very low (e.g., less than 1 frame/s), the content tends to be rendered before anything has arrived from the server, leading to empty or very low quality frames.

At very low frame rates (e.g., for a slide show), the content of each frame dynamically refreshes as more data arrives from the server. In generally, however, to experience the full quality of what a JPIP server has delivered to you, it is necessary to navigate backwards and forwards over the frames, or to play the video repeatedly. In a future version of "kdu_show", this will of course no longer be necessary, but JPIP browsing will always have the feature that the quality of experience increases each time you revisit the same frame or region of interest.

6 Properties, Metadata, Overlays and Editing

You may use "*kdu_show*" to view the coding parameter attributes and other properties of the codestream (or codestreams). You can also display, add and edit metadata in a JP2 or JPX file, resaving edited metadata in the JPX format – only a limited subset of metadata can be saved in the more restrictive JP2 format. If the content is being served dynamically via JPIP, metadata can be viewed, but cannot currently be edited.

6.1 Viewing Codestream Properties

Use the File->Properties menu item or the "ctrl-p"/"cmd-p" accelerator to view the coding attributes and other properties of the current codestream. Where more than one codestream is involved in the current image display, the application chooses one of them to display its attributes. Codestream properties include comment marker segments, coding parameter marker segments and tiling attributes.

You may click on any attribute to read a detailed description of that attribute's interpretation, as provided by the Kakadu core system.

6.2 Viewing the File Structure with Metashow

Additional attributes are provided by the file format in which codestreams are embedded. To examine the file structure itself, use the Metadata menu or the "ctrl-m"/"cmd-m" accelerator to activate the "*metashow*" tool. You may click on various nodes within the metashow tool's tree view to obtain additional details or to have the main window navigate to corresponding image elements. For example, clicking on the second compositing layer header box will cause the main window to enter single compositing layer mode and display the second compositing layer. Similarly, clicking on a contiguous codestream box or codestream header box will cause the main application to enter single component mode, while clicking on the composition box, if any, will cause the application to enter the composited image mode.

Clicking on some leaf nodes in the metadata tree will cause their contents to be expanded as text, while for other boxes a generic hex dump is provided.

If you select a metadata item which corresponds to an editable JP2/JPX metadata node, the "*Open in Metadata Editor*" button will be activated, allowing you to edit the node, add descendants, etc. Although edited metadata does not become part of the file structure until the file is saved and reloaded, "*metashow*" uses strike-through conventions to highlight elements in the file structure which have been deleted (solid strike through) or changed (dashed strike through) by the editor. These features may vary slightly between "*kdu_winshow*" and "*kdu_macshow*".

6.3 Adding and Editing Metadata

To add metadata use the Metadata menu or the "ctrl-a"/"cmd-a" accelerator. This opens up the metadata editing dialog, which allows you to enter text labels or to import externally edited label or XML box contents. If no focus box has been selected, the metadata will be associated by default with the top-most current image entity (compositing layer or codestream), depending on the current image display mode (see Section 2.5).

If a focus box was defined at the time when you hit the "ctrl-a"/"cmd-a" accelerator, the metadata will be associated by default with the top-most codestream which overlaps with the focus region; it will also be associated explicitly with that region. These associations are achieved by embedding appropriate number list and ROI description boxes in the JPX file (when you save the edited result).

You may alter any of the above associations within the metadata editing dialog. The editor explicitly excludes options which are illegal; for example, metadata associated with a focus region must also be associated with at least one codestream, not just with compositing layers. You will see that you can add multiple associations, but it is up to you to ensure that this is meaningful.

You can edit metadata that you have added yourself, or metadata that existed in the file already, in any manner you like. "*kdu_show*" provides many mechanisms for editing metadata. Some types of metadata can be edited directly from the metadata catalog sidebar, discussed in Section **??**, but all metadata can be edited from the metadata editor window. By and large, the metadata editor is launched whenever you *right-click / ctrl-click* on a relevant item in the image view or in the metadata catalog sidebar. The Metadata menu and the "ctrl-e"/"cmd-e" accelerator can also be used. You can add descendant nodes via the "*Add child*" button, which initially creates a text label box, but you can change the box type. Short text labels can be entered directly, or you can launch a separate editor for text or other box-types – for example, you might launch the Dashcode application to edit an XML node on an Apple system. An important type of metadata node is a "*link*" (or "cross-reference") to another metadata node. You can create links by pasting the target of the link to the pastebar (appears above the metadata catalog sidebar) using the "*Copy to paste bar as link*" button and then converting another node (typically one you have just added using "Add child") to reference the link by clicking the "*Replace using paste bar*" button.

You will find that before "kdu_show" adds links to the metadata, it asks you one or more questions about what type of link you would like to create. You may find this confusing at first, in which case you are encouraged to read the more detailed discussion of links in Section **??**.

You can freely move around the metadata hierarchy by using the navigation buttons provided: "*up to parent*"; "*view descendants*"; "*prev*" and "*next*". These allow you to descend into the children of a node (view descendants), navigate between siblings (prev and next) and go back up to the parent of a node (up to parent). Along the way, you can "Delete" nodes as well. You can even grow the metadata hierarchy from above your current editing position by using the "*insert parent*" button.

It is worth noting that all relevant changes made within the metadata editor are reflected to the metadata catalog sidebar described in Section **??**, the image view and even the metashow window (as described above, metashow only reflects changes to data that exists in the actual file). You can also use the "*Find in metashow*" and "*Find in catalog*" buttons to have the node you are currently editing selected within the metashow or metadata catalog sidebar.

When editing metadata, we strongly recommend that you regularly save your work – see Section 6. The reason for this is that the metadata editor does not contain any undo functionality to recover from an unfortunate editing mistake. The only exception to this is the interactive shape editor, which does keep a history of changes to regions of interest and allows you to back up or forward through this history using the familiar undo/redo operations – see the Metadata menu for information on the accelerator keys you can use to undo or redo a shape editing step. The shape editor is described in much more detail in Section **??**.

6.4 Changing Image Entity Associations

Although this topic formally belongs in the previous sub-section, we give it special treatment here so as to draw attention to some of the possible pitfalls of manipulating image entity associations within the metadata editor. Image entity associations are controlled via the upper-right hand pane within the metadata editor window. Image entities can be any of the following:

Codestreams: A JPX source may contain any number of codestreams and metadata may be associated with these codestreams. As explained below, if your intention is to associate metadata with an image element as a whole, we recommend using compositing layer associations instead. However, spatial region of interest metadata must be associated with codestreams one way or another in order to give meaning to the coordinates that are used. Although it is formally sufficient to associate regions of interest only with codestreams, we recommend that you also explicitly provide the compositing layer associations for the region of interest, as explained below. This allows you to identify the specific use of a codestream within a composition or animation, to which the association applies.

Compositing layers: JPX compositing layers may be formed from multiple codestreams. More importantly, though, any number of JPX compositing layers may refer to the same codestream. This allows you to re-use imagery content in different parts of a composition or animation, while labeling the specific use with a unique composition layer index. Individual compositing layers can also be re-used within JPX files, but then you have no way to associate metadata with a specific use.

Considering that compositing layer associations allow you to distinguish between different uses of the same codestream, **we strongly recommend**

always incorporating one or more compositing layer indices in the collection of associations that you assign, even if this is not strictly necessary. Some of the navigation and animation machinery with "kdu_show" works much better if compositing layer associations are available, because codestream associations alone are insufficient for unambiguous interpretation of a navigation target without parsing all compositing layers in an attempt to discover how each codestream is used.

Rendered result: You can associate metadata with a somewhat ambiguous entity known as the "rendered result." Presumably, the JPX file format defines this entity with the intention of referring to a complete composited image, potentially formed from many compositing layers. However, if your content involves multiple composited frames (i.e., animation), it is not clear how useful this type of association is. Moreover, there is not a huge semantic difference between associating metadata with the rendered result and not associating metadata with anything at all. In general, we suggest that you do not create content using this option, but it is there so that you can interpret existing content.

Any combination of the above associations is permitted and any number of codestreams or compositing layers may be listed. Where there are multiple compositing layers or codestreams listed, "kdu_show" uses this information to assist with intelligent navigation. For example, if an associated metadata item is double-clicked in the metadata catalog sidebar, the image view will automatically be navigated to the composited frame that best matches the set of associations. If there are multiple frames that contain compositing layers with which the metadata is associated, the one that contains the most such matches, if any, is the target of this navigation process. You can of course remove all image entity associations.

You should note that image entity associations are created via JPX number list boxes. Kakadu does not needlessly create a new number list box for each metadata item that you wish to associate with image entities. Instead, it re-uses existing number list boxes wherever possible, so that all metadata associated with a particular collection of image entities sits underneath the number list box. Using the "*up to parent*" button, the metadata editor allows you to navigate all the way up to the number list box itself – notice that "*nlst*" appears within the "*Box-type (4cc)*" field on the left side of the editor window. Editing the image entity associations or making any other permitted changes at this level may have unintended consequences, because it will affect the associations of all metadata that is currently descended from the number list box. In the special case where all image entity associations are removed directly from a number list box, the number list box is deleted and all of its contents are moved to the top level of the file. This will leave the editor displaying a Box-type of "root".

Considering the far reaching consequences of editing number list boxes directly, **you should normally edit the associations only of the top-level metadata nodes within a number list box** – i.e., the ones you encounter immediately before the "nlst" box itself while using the "up to parent" navigation button. In this case, changing the image entity associations causes the metadata to be moved to a new number list box, without affecting the existing one. If all metadata nodes are moved out of a number list box in this way, the number list box itself goes away automatically, since this is usually desirable.

As has already been mentioned, if you hit the "ctrl-a"/"cmd-a" accelerator while the mouse cursor is positioned within the image view, with no focus box selected, a number list node will be created to represent the most relevant image entity (compositing layer or codestream), unless one already exists, in which case the metadata editor will open at the existing number list node. As mentioned, this can be a dangerous place to edit associations, so we recommend that you first click the "*Add child*" button to add a new descendant of the number list node. At that point, it is completely safe to change the image entity associations – **this is the only safe way, for example, to create a new top-level metadata item that has not image entity associations**.

6.5 Interactive Shape Editor

The metadata editor contains many additional features, that you will discover as you use it, but one particularly powerful feature is the "*Interactive Shape Editor.*" If you have just added a region of interest by hitting the "ctrl-a"/"cmd-a" accelerator while a focus box was defined, the shape editor is activated immediately; you will notice a white outline with yellow circles around the region vertices within the image view. You can close the shape editor re-open it at any time by clicking the relevant buttons within the metadata editor. During shape editing, the metadata editor window and the image view work collaboratively. In the simplest case, your region of interest may be a single rectangle or a single ellipse whose major and minor axes are oriented in the horizontal and vertical directions. However, you can readily drag the vertices of these shapes around in the image view – try clicking the "*Apply*" button after doing so. You can always revert to a simple rectangle or ellipse, or even remove region associations altogether by checking the "*No ROI*" option. The interactive shape editor panel within the metadata editor provides you with a lot of functionality. You can "*Add*" additional quadrilaterals or ellipses to form a composite region – the "ctrl-a"/"cmd-a" accelerators also do this from within the image view. You can also "*Delete*" quadrilaterals or ellipses from a composite region. Just where things are added and what is deleted depends upon the edge that is selected within the image view.

You will notice that the shape editor provides three different editing modes. The most natural is the "*Vertex mode*", in which you manipulate the region by selecting and dragging vertices around; each time you click on a vertex the selected edge also changes, with appropriate visual cues. The "*Skeleton mode*" is similar except that you select and drag edges around. The "*Path*" mode may be used to define and manipulate quadrilateral regions that form path segments – typically they are long and thing. In this mode, the width of the path segments may be automatically fixed by clicking the "*Fix path width*" button.

In most cases, it is best to define paths initially with a fixed width of 1 (the smallest possible value) and then open out the entire path if desired by changing the width later on. By dragging path end-points and using the "*Add*" button or "ctrl-a"/"cmd-a" accelerator, you can rapidly draw a path around a polygonal path over the image view. While all this is going on, you can navigate freely in the image view using the regular mechanisms, zooming, panning, rotating, etc.

Paths are useful in a number of ways. You may be interested in defining a composite region of interest that actually is a narrow path of some fixed width. In this case, you might can even create rounded corners for your path by checking the "*+junctions*" option during path definition – each time you add a new segment an ellipse is also inserted, whose location and dimensions are determined so as to provide you with rounded corners. Releasing and then fixing the path width to a different value automatically adjusts everything, including your rounded vertices. Alternatively, you may use paths to define a polygonal region and then click "*Fill path*" to have the editor automatically build a composite region that consists of non-overlapping quadrilaterals. In the end, all JPX regions of interests are collections of quadrilaterals and ellipses, one way or another.

The shape editor also allows you to define arbitrary regions of interest using a free-form input style that is accessed via the "*Start scribble*" button. Typically, you scribble in a path or region boundary and then click the conversion button, which is one of "*Scribble -> path*" or "*Scribble -> region*", depending on the current shape editing mode. Converted scribble can consume a lot of quadrilaterals, for which there is a fundamental upper limit in the JPEG2000 standard. For this reason, the conversion process involves approximation. You can set the level of approximation via the supplied slider and observe how close you are to the maximum complexity via the "*shape complexity meter*."

You should note that all spatial dimensions shown in the metadata editor correspond to locations on the relevant codestream's high resolution grid, measured relative to the upper left hand corner (0,0) at which the codestream's imagery is placed on that grid. In most cases, image pixels correspond to high resolution grid points, so that regions of interest are defined at pixel precision. However, JPEG2000 codestreams are readily constructed in which the actual image pixels occupy multiple high resolution grid points[1]. In this case, regions of interest are defined at sub-pixel precision, which can be useful in some circumstances. The important thing to note here is that the dimensions reported, or set using the "Fix path width" button, for example, are independent of the scale being used in the image view (or the orientation of the view for that matter). You may, for example, need to zoom into the view considerably to see a path with a very small width.

[1]The number of grid points per pixel in each direction corresponds to the sub-sampling factors that are controlled by the 'Ssampling' codestream parameter attribute. You can, of course, supply different sub-sampling factors for each image component (e.g., colour plane) of an image during compression, but none of them has to be equal to 1.

6.6 Metadata Overlays

Once you have some spatially associated metadata, you will notice that "*kdu_show*" highlights its presence via partially transparent overlays; by default these flash through a set of visual states – internally all of this is implemented by the platform independent 'kdu_region_compositor' object that is highly customizable. You may use the Metadata menu to control the appearance of the overlays. Most usefully, the "ctrl-alt-t"/"cmd-opt-t" accelerator may be used to toggle the overlay mode between *flashing*, *static*, and *no overlays*. You may also control the heaviness of overlays via the Metadata menu. Finally, you may control the minimum size of elements which will appear on overlays.

In the flashing or static overlay mode, when the mouse cursor is positioned over a region of interest, various actions can be triggered based on the associated metadata. The "ctrl-e"/"cmd-e" accelerator opens the metadata editor at the region of interest node, as does "ctrl-click" or "right-click". The opening of the metadata editor at a region of interest node in turn automatically launches the interactive shape editor. Simply left clicking on a region of interest, as displayed by overlays in the image view, causes the most logically associated item in the metadata sidebar catalog to be selected automatically – see Section **??** for a discussion of metadata logic, as it relates to the catalog sidebar.

Double clicking a region of interest revealed by the overlay, not only causes the most logically associated item in the metadata sidebar catalog to be selected, but also causes a tight focus box to be placed around the region of interest. This is of particular interest during remote image browsing via JPIP, because the focus box causes all of the server's bandwidth to be dedicated to sending quality improvements for the content within the focus box.

While the image view has focus and the mouse is over a region of interest, as revealed by the overlay mode, hitting the "ctrl-L"/"cmd-L" accelerator key causes a link to the region of interest node (actually a cross-reference to the JPX ROI description box) to be temporarily recorded within the catalog pastebar – appears above the catalog sidebar on the right hand side of the image view. This link can subsequently be inserted into the metadata hierarchy, either by hitting the "ctrl-v"/"cmd-v" accelerator (inserts the link as a descendant of any item currently selected in the metadata catalog), or by using the "*Replace using pastebar*" button within the metadata editor.

You will find that before "kdu_show" adds links to the metadata, it asks you one or more questions about what type of link you would like to create. You may find this confusing at first, in which case you are encouraged to read the more detailed discussion of links in Section **??**.

7 The Metadata Catalog Sidebar

When you open a JPX image with associated metadata in the form of text labels (JPX label boxes), regions of interest (JPX ROI description boxes) or references to image entities (JPX number list boxes) an auxiliary catalog panel is automatically opened on the right hand side of the image view. The same thing happens when you first add such metadata to an image which previously had none. You can always hide the catalog panel via the Metadata menu, but it is very convenient to keep around. The catalog contains a hierarchically organized record of all textual metadata in the source, under three broad categories:

Structured metadata: This category contains all metadata nodes that sit at the top level of the file, other than regions of interest and image entity references, along with all their descendants. Typically, the top level entries under this category are text labels that are independent of specific regions of interest or image entities. Examples, might include a street directory, a list of keywords, a list of topics of interest, generic information about the content, or perhaps hyperlinks to other images or web-pages. Top level entries in this category might also be links to other nodes that may lie anywhere in the metadata hierarchy. The descendants of these items will often also be textual metadata or links, but they may include regions of interest and/or image entities, which themselves may have descendants.

Top/orphan image associations: This category contains all top-level image entity references (JPX number list boxes) and their descendants. An image entity usually refers to a specific codestream or a specific compositing layer within the source, but it may refer jointly to a collection such things[2]. A bare image entity reference that contains no descendants will show up in the metadata catalog as "<imagery>", in a green coloured font. You can add such a thing to an image that contains no metadata at all, simply by clicking anywhere in the image view (don't define a focus box) and then hitting the "ctrl-a"/"cmd-a" accelerator. This will create a reference to the compositing layer associated with the location in the image view where the mouse was located (often layer 0), but it may create a codestream association if you were viewing individual codestream image components. Once an image entity reference has textual metadata as descendants, however, you will notice that the uninformative "<imagery>" identifier is replaced by these descendants, but they continue to use a green font. It follows that the entire Top/orphan image associations category within the sidebar can usually be expected to appear in green, since everything there is either an image entity reference or one of its descendants. More precisely, the green font is reserved for all metadata nodes that are associated, either directly or through their ancestry within the hierarchy, with image entities as a whole (as opposed to spatial regions of interest). Image entity references and their descendants that appear as descendants of nodes in the "Structured metadata" category will also appear in green, but of course they cannot appear at the top level of this category. This is an important visual cue, because double-clicking a green item navigates

[2]Image entities references may also refer to a poorly defined entity called the "rendered result" in the JPEG2000 Part2 standard. Ultimately, a number list box is just a collection of integers, one bit of which distinguishes codestream references from compositing layer references, while a special value signifies the rendered result.

the image view to the relevant image entity, no matter where that item appears in the catalog.

Top/orphan region associations: As you may now guess, this category contains all top-level region of interest nodes (JPX ROI description boxes) and their descendants[3]. All of these are displayed with a red coloured font. Bare region of interest nodes, having no textual descendants, are identified within the catalog by the string "<region (ROI)>". Once the node has textual metadata as descendants, however, you will notice that this uninformative string is replaced by these descendants, which will retain the red coloured font. As with the green nodes, the red font is reserved for all metadata nodes that are associated, either directly or through their ancestry within the hierarchy, with a region of interest. These may potentially appear within the "Top/orphan image associations" category (but not at its top level) or within the "Structured metadata" category (but not at its top level). The red colour is an important visual cue, because double-clicking a red item within the catalog navigates the image view to the relevant image entity and places a tight focus box around the region of interest. The act of selecting a red item within the catalog causes the metadata overlay imagery (if enabled) to momentarily flash so as to highlight the region of interest and hide all other regions of interest for a short while.

7.1 Links within the Catalog

We have already mentioned the possibility of links (or cross-references) from one point of the metadata hierarchy to another. Without links, the metadata structure would only be able to express parent-child type relationships. Specifically, it is natural to think of a node's descendants as being associated with it in some semantic way, although the descendants might not have any direct association with each other. For example, a text label node "Buildings" may have descendants "Gymnasium" and "Hospital", both of which are clearly associated with the category of "Buildings", but have no other obvious association with each other. This parent-child relationship is exploited to associate metadata with image entities or with spatial regions of interest, in the manner described above – this is what leads to the use of green and red colour codes for items in the metadata catalog.

Links allow for the incorporation of richer semantics than those that can be expressed using hierarchical parent-child relationships alone. To fully un-

[3]The notion of "top-level" in this case may be a little misleading. For a JPX ROI description box to be meaningful, it almost invariably has to be a descendant of a number list box. So what we mean by top-level here is that the JPX ROI description box is an immediate descendant of a top-level number list box. If all of a number list box's descendants are ROI description boxes, it will not have any representation in the "Top/orphan image associations" category. However, if it also has textual or other immediate descendants that can be represented in the catelog (i.e., these are not descended from regions of interest) then these will show up in the usual green font within the "Top/orphan image associations" category.

derstand how Kakadu constructs and interprets links using JPX cross-reference boxes, you are referred to the Kakadu API documentation. For the moment, however, this section serves to provide you with a quick introduction to the three types of link semantics supported by Kakadu and how they are represented within the catalog. The three types of links are as follows:

Alternate child links: This is probably the most natural type of link and also arises from the most obvious way of using JPX cross-reference boxes. Alternate child links to metadata nodes that themselves can be represented within the catalog appear with the same text as the link's target, but with a blue coloured italic font and solid underlining (as one might expect to see hyperlinks displayed in a web-browser).

Semantically, the target of an alternate child link is a metadata node that could logically appear at the same location as the link. That is, the target of the alternate child link could logically be considered as a child of the linking node's own parent and all of the target's descendants could logically be considered descendants of the linking node's parent.

The reason for using alternate child links is that they effectively allow metadata to be associated with (or at least, discovered from) multiple parents. For example, if "Buildings" and "Sports Facilities" are two top-level text label nodes in the metadata hierarchy, it is clear that "Gymnasium" could logically be considered as a descendent of either "Buildings" or "Sports Facilities". From a mathematical perspective, one could say that there exist partial orderings (or containment relationships) of the form

$$\begin{aligned} \text{"Gymnasium"} &\prec \text{"Sports Facilities"} \\ \text{"Gymnasium"} &\prec \text{"Buildings"} \end{aligned}$$

but there is no such relationship between "Buildings" and "Sports Facilities". One can represent these types of semantic relationships by making "Gymnasium" a descendant of one of its logical parents (or containers) and incorporating an alternate child link to "Gymnasium" as a descendant of its other logical parents.

Alternate parent links: In the same way that alternate child links provide a means of discovering metadata that is logically contained within other nodes, alternate parent links provide an explicit means of discovering the containers. Internally, whenever Kakadu discovers a link, it also keeps track of where the link came from and records this information with the link target. Considering this, an alternate child link would appear to be sufficient to allow navigation from a node to its alternate children and back again from alternate child nodes to their alternate parents.

However, this implicit mechanism for discovering alternate parents (alternate logical containers) cannot work in general during remote browsing with JPIP. This is because JPIP servers generally deliver only those portions of the metadata hierarchy that are relevant to the requests received. Suppose an interactive user is currently inspecting a certain region of an image (defined by the focus box). The server will at some point deliver the metadata that is associated with that region, because "kdu_show" asks for metadata that is associated with the window of interest. Since the underlying file structure fundamentally embeds metadata nodes within their containers (parents), it follows that the client (the viewer or browser) must be able to discover the parent of any metadata node that is delivered, and hence walk up/down the hierarchy. However, there is no way to discover an alternate parent, if the alternate child link has not been delivered by the server. Moreover, there is no reason for a JPIP server to deliver nodes that link to metadata nodes of interest.

To avoid these problems, Kakadu treats certain cross-reference boxes within the JPX metadata structure as explicit alternate parent links. Formally, an alternate parent link is a cross-reference box that references the alternate parent's metadata in isolation, rather than the sub-tree that has the alternate parent as its root. Exactly how this is done is explained in the Kakadu API documentation, but all you need to know here is that alternate child and alternate parent links can never be confused with each other. Moreover, the use of alternate child and alternate parent links does not create any kind of recursive containment dilemma, because an alternate child link belongs to the sub-tree rooted at the alternate parent, whereas the alternate parent link cross references the parent metadata in isolation, rather than its sub-tree.

For full discovery of alternate containment relationships during remote browsing, allowing for efficient navigation into alternate children and back up to alternate parents, it is strongly recommended that you create alternate child and alternate parent links in pairs. When you create link nodes, using any of the methods described in Sections **??**, **??** or **??**, you are given the option to create both types of links together. In most cases it is advisable to do this. There are, however, some cases in which you might feel it is sufficient to incorporate only the alternate child or only the alternate parent link, based on the semantics of your metadata.

Within the catalog, alternate parent links show up with a magenta coloured italic font, with dashed underlining (underline patterns might differ between "kdu_winshow" and "kdu_macshow"), but everything else should be the same in both viewers. The alternate parent link appears as a descendant of the node whose alternate parent it is describing, but this winds up looking fairly natural. Of course, the way the "kdu_show" application displays metadata is independent of the JPEG2000 standard and independent of the semantic interpretation offered and supported by Kakadu. Another viewer, for example, might show alternate parent links in a drop-down box when the child is clicked in some way.

Grouping link nodes: The most difficult type of link to understand is the grouping link, but it serves a very useful purpose. The distinguishing feature of grouping link nodes is that they can have their own descendants – in fact, that is how they are identified within the JPX metadata hierarchy. Whenever you need to link a node that has its own descendants directly to some other element in the metadata hierarchy, you must use a grouping link.

Semantically, all grouping link references to a given target can be considered to belong to the same logical group, even though they may have different parents and different descendants. One good way to use grouping links is to construct multi-lingual metadata. To this end, one can create a few top-level text label nodes such as "English", "German", "Mandarin", etc. Then, at any point in the metadata hierarchy where you wish to include multiple versions of the metadata for each language, you can insert a grouping link to the relevant language label and then add the descendants, scripted in the appropriate language.

To make the above example clearer, suppose you have a collection of regions of interest defined within an image. Under each region of interest, you can add a grouping link to the relevant language, then under the language link you can add language-specific metadata describing the region of interest. You can, of course, then incorporate language-specific links from these descendants to other language-specific portions of the metadata hierarchy. Within the catalog, all of the grouping links created in this way will naturally appear at the top level of the "Top/orphan region associations" category. Since all grouping links to "English", for example, have the same appearance and belong to the same group, there will only be one entry for "English" Top/orphan region associations within the catalog, under which all the English scripted descendants of all regions of interest will appear. The same goes for the other languages.

Of course, grouping links can be used for things other than language associations and there is nothing about "kdu_show" or its catalog structure that specifically targets the idea of multi-lingual associations. It is possible to chain groups together by arranging for a grouping link to point to a node that itself is a grouping link to another node, so long as the chain ultimately ends with a non-grouping link node. The target of a grouping link might be an isolated label or it might have its own descendants. These descendants might include links (typically alternate child links) back to members of the group, which could help with discovery in some browsing situations.

In "kdu_show", grouping links are displayed using an orange coloured italic font with solid underlining. Unlike alternate child and alternate parent links, grouping links are not displayed with an upward or downward pointing arrow, since they neither point up nor down within any kind of logical containment – you could think of them as sideways links, rather than alternate up/down links.

7.2 External hyperlinks within the Catalog

The metadata catalog possesses some primitive ability to derive external hyperlinks from text labels. Specifically, if a text label commences with one of the recognizable protocol strings "http://" or "jpip://", or if it appears to refer to a JPEG2000 file, the label is displayed in a light blue italic font, with underlining. Double clicking on such an item in the catalog will either open the default web-browser for your system or open another window within the "kdu_show" application, allowing you to view the referenced local or remotely browsed JPEG2000 image. Actions are taken based on the most natural interpretation of a reference. For example, a label of the form

./test_file.jpx

is interpreted as a reference to an image file at the same level of the file system as the current one. If the current file is being browsed remotely via JPIP, the reference will be converted to an appropriate JPIP URL, derived from the current one. On the other hand, a label of the form

jpip://server.com/test_file.jpx

will always be treated as a JPIP URL and opened, if possible, when the link is double-clicked.

In the future, it makes sense to derive external hyperlinks from appropriate XML boxes within the metadata, but this has not yet been implemented in the "kdu_show" demo application. Kakadu itself, of course, does fully support XML metadata.

7.3 Editing directly within the Catalog

Some simple editing operations may be performed directly within the metadata catalog sidebar, so long as the metadata editor dialog is not open at the same time. As with all editing operations, this is only currently possible for local files, as opposed to content served up remotely via JPIP.

Specifically, you can copy or cut items to the pastebar and you can paste such items back into the catalog as descendants of any given node. You can also delete items directly from the metadata catalog by pressing the delete key or the backspace key (so long as the catalog has focus, as opposed to the image view). These operations can all be performed either with accelerator keys or via the Metadata menu.

Directly copying an item from the catalog to the pastebar may be achieved using the "ctrl-c"/"cmd-c" accelerator key while the metadata item of interest is selected within the catalog; this works only for text labels. Subsequent pasting of directly copied text labels back into the catalog, using "ctrl-v"/"cmd-v" accelerators in the usual way, creates a duplicate copy of the text itself, as a descendant of the node currently selected in the catalog. Descendants of the copied node are not themselves copied. There is a way to copy a metadata node along with all of its descendants, which can be accessed via the "Copy descendants" button in the metadata editor; this launches a potentially complex process, which not only copies descendants but also rebuilds links that exist within the copied content so that copied links whose targets are also being copied are modified to point to the copied targets. We have used this method very successfully to generate multi-lingual metadata, by copying a complex metadata structure created in one language and then translating each element in the copy.

Alternatively, you can use the "ctrl-L"/"cmd-L" accelerator to copy a selected item from the catalog to the pastebar as a link. Items copied as links show up in the pastebar with the same font (italic, blue, underlined) as alternate child links in the catalog. You can also create links to other metadata nodes in the pastebar by using the "*Copy to paste bar as link*" button in the metadata editor, or by using the "ctrl-L"/"cmd-L" while the image view has focus and the mouse pointer is over a region of interest, as revealed by metadata overlays. When you paste a link from the pastebar back into the catalog, using the "ctrl-v"/"cmd-v" accelerator key or via the Metadata menu, a link node is created as a descendant of the currently selected item in the catalog. See Section **??** for more information on link semantics that will help you understand how to choose from the options that are presented to you.

You can use the "ctrl-x"/"cmd-x" accelerator key, or the Metadata menu, to "cut" an item from the metadata catalog to the pastebar. This shows up in the pastebar using a strike-through textual presentation. It is also possible to "cut" region of interest nodes directly to the pastebar by hitting the "ctrl-x"/"cmd-x" accelerator while the image view has focus and the mouse pointer is located within a spatial region of interest, as revealed by metadata overlays. You can, of course, clear the "cut" operation from the pastebar without having any effect, by clicking on the small "x" button to the right of the pastebar. Otherwise, the next attempt to paste from the pastebar (either in the metadata editor or while using the "ctrl-v"/"cmd-v" accelerator within the catalog sidebar) will move the metadata node in question, along with all of its descendants, to the new location.

7.4 Logical Associations within the Catalog

The metadata catalog sidebar and the image view itself work together in a collaborative fashion to help you navigate through complex sources. This can be especially useful during remote content browsing via JPIP, but it also of considerable interest for local files that have a rich structure. Kakadu is designed to be able to handle large amounts of metadata efficiently and also to handle huge images (and many of them) efficiently.

While browsing within the image view (e.g., panning, zooming, moving from frame to frame, etc.), metadata overlays reveal spatial regions of interest and clicking within these regions causes associated metadata to be selected automatically within the catalog. Similarly, clicking any point on an image surface that is not contained within spatial overlay data causes metadata associated with the relevant image entity to be selected automatically within the catalog.

You might then use the catalog to discover other related metadata, with other imagery associations; double clicking on items of interest in the catalog will automatically navigate the image view to the relevant associated content. As already noted, the act of selecting an item in the metadata catalog causes any region associations to be revealed for you in the image view. Added to all this, based on identified metadata of interest, dynamic animations can be constructed to walk through larger collections of imagery and metadata automatically, as described in Section **??**.

An important question that arises is what action to take when there are multiple metadata items associated with a region of interest or image entity selected within the image view. What "kdu_show" does is to walk through all metadata within the catalog that can be logically shown to have an association with the relevant image content, evaluating a "distance measure" on the logical association graph. Logical associations are formed through regular parent-child containment relationships, by following links and even by following links in the reverse direction. For details of Kakadu's association logic sub-system, you are referred to the Kakadu API documentation; the system is highly flexible.

What matters here is that the results of logical analysis produce some number of "most closely associated" elements in the metadata catalog. If there is more than one candidate, Kakadu decides amongst these candidates by evaluating the logical associations of each candidate with the most recently selected item within the metadata catalog sidebar. This can be very useful. A good example would be that of multi-lingual metadata, which we have already used for illustration purposes in Section **??**. Suppose a region of interest is associated with many text labels within the metadata catalog, all appearing at the same level of containment with respect to the region of interest node itself, but each represented in a different language. If the last selected item in the catalog was expressed in French, for example, "kdu_show" will generally be able to establish a logical connection between it and the French label for the selected region of interest, by following links (including following them in reverse), as well as parent/child relationships. To make sure that this works, you should use grouping links in your metadata to definitively associate French language sub-trees of the metadata hierarchy with a single node (perhaps one that reads "Francais").

It also happens (perhaps very frequently) that multiple image entities or regions of interest that might be displayed in the image view are associated with a given item with the metadata catalog sidebar. You can use this to your advantage to arrange for all associated image regions to be highlighted within the metadata overlay. This happens momentarily after you select an item in the catalog. To continuously restrict the overlay content that you see to data that is associated with a selected element in the catalog, you can use the Metadata menu or the "ctrl-r"/"cmd-r" accelerator. Hitting the accelerator a second time releases the overlay restriction. These overlay highlighting and restriction mechanisms are all based on Kakadu's sophisticated association logic sub-system, so good things generally happen.

When you double-click an item in the metadata catalog (or press enter while the item is selected and the catalog is in focus), the most immediately associated region of interest or image entity is selected within the image view, which is achieved by placing a focus box around the item. If there are multiple regions or image entities that are all associated equally with the selected catalog item, each time you perform this action the image view should change to reveal the next equally associated entity. This can also be a very useful behaviour for some complex sources.

7.5 Metadata-Driven Animations

One very interesting feature of the metadata catalog sidebar is the ability to construct and play novel animations dynamically from the metadata itself. The easiest way to access this functionality is to hold the shift key down while double-clicking on an item of interest within the metadata catalog. In "kdu_macshow", you can also tripple-click on the item of interest. On both platforms, alternate ways to invoke the feature include holding the shift key down while pressing the enter key, with the item of interest selected in the catalog (so long as the catalog has focus, as opposed to the image view).

Metadata animations are constructed whenever the above actions effectively identify more than one spatial region of interest and/or more than one compositing layer within the JPX source. Region of interest and image entity metadata are effectively identified if they belong to the selected item, if they are descendants of the selected item, if they are linked by the selected item or its descendants, or if they are the most directly associated region or image entity for the targets of such links. Together, these cover pretty much all the semantic memberships of interest.

Metadata-driven animations consist of a sequence of frames, where each frame corresponds to a distinct item that has been effectively identified by the invoking action, as described above. In some cases, these frames might correspond to distinct composited frames within a JPX animation, in which case the frames are visited with the usual dynamics (except that the metadata may be selecting only a subset of the complete animation).

In other cases, the metadata-driven animation may consist of a sequence of regions of interest within a single image or composited frame. In such cases, the animation is instantiated to use special dynamics in which the view is appropriately zoomed and then automatically panned around as the animation moves from region to region. You can, of course, still zoom, rotate, resize the view and so forth while the animation is going on, but you will not be able to manually pan the view if it is being controlled by a metadata-driven animation that involves regions of interest.

Combinations of the above cases also occur and can be of great interest. For example, the elements of the metadata-driven animation might correspond to distinct composited frames of a regular animation, but the metadata may identify a separate region of interest in each frame, in which case the view automatically keeps the region of interest centred (if possible) while the animation is played.

It can happen that a source file does not offer any naturally animated content, so that no animation bar normally appears at the bottom of the view window – for example, the content might consist of a single image or composited frame. Nevertheless, metadata-driven animations may still be synthesized from appropriate metadata, in which case the animation bar appears for the duration of the dynamic animation.

The animation bar, the Play menu and relevant accelerators may be used to control metadata-driven animations in the same way as regular animations or video. Specifically, you can turn the repeat mode on or off, you can change the playback direction and you can alter the playback speed. One big difference with metadata-driven animations, however, is that they are automatically terminated if you click inside the metadata catalog sidebar. This is useful, because it is often a way of starting a new animation or selecting an item that has come to your attention as being of particular interest.

While a metadata-driven animation is in progress, the metadata items that are responsible for inducing each frame are selected automatically (one by one as the frames go by) within the metadata catalog. At slow animation rates, this can be very useful. At higher frame rates, this selection process occurs only intermittently (several times per second).

Metadata-driven animations also work with JPIP for remotely browsed content.

7.6 How the Catalog works with JPIP

The catalog panel machinery updates itself automatically whenever metadata is added, deleted or modified via the metadata editor. Similarly, during a remote browsing session with JPIP, the catalog updates itself automatically as new metadata is found in the cache, having been sent by a JPIP server. It can be quite a gratifying experience to watch the catalog expand and rearrange itself as data arrives, all the while maintaining the location and expansion properties of any currently selected item.

Items in the catalog that are known to exist but do not yet have sufficient data to be displayed are given temporary names involving ellipses. Links that are known to exist, but for which the link target is not yet available are also given temporary names of the form "<link pending>". Region of interest nodes or number list nodes whose descendants have not yet arrived are displayed as "<imagery>" or "region (ROI)>", until renderable text content arrives to replace these temporary names.

The "kdu_show" application uses the Kakadu API capabilities effectively to generate explicit requests for metadata from the server, as and when this becomes appropriate. In the interest of communication efficiency, descendants of an item in the catalog are not explicitly requested unless you attempt to expand the item into its descendants. Similarly, explicit requests for the target of a link node that has not yet been resolved are only generated if the link is expanded within the catalog view.

Response times can be quite low, unless the server is heavily loaded, because Kakadu's JPIP client implementation provides a special mechanism to pose what are called "out-of-band" requests for small amounts of additional content, while the server continues to stream responses to the window of interest (or windows of interest). However, if you have a lot of windows open, each with their own active windows of interest being used to fetch data from the server, you are likely to run out of physical JPIP channels, so that the out-of-band requests must be interleaved onto the least loaded active channel.

It is worth noting that active view windows (ones for which not all the content is available yet) automatically generate their own JPIP window of interest (WOI) requests, potentially restricted by a prevailing focus box. These requests also ask the server to deliver any metadata that is immediately associated with the WOI (e.g., with the focus region or the window viewport). These requests are posed and, if necessary, updated in such a way that the server is not required to deliver metadata that cannot be rendered within the catalog and it is not required to deliver descendants of any such metadata items. Requests for those things are generated as you navigate within the metadata catalog itself.

Finally, the metadata required to define a metadata-driven animation (see Section **??**) is itself automatically requested by the animation machinery if not currently available within the cache. What this means is that you can shift-double-click on an item in the metadata catalog and watch as the relevant associated metadata required to discover a metadata-driven animation is retrieved, then watch as the animation is realized from the newly discovered metadata. These two activities are actually interleaved, since Kakadu's 'kdu_region_animator' object itself generates the required metadata requests on demand, so the animation can commence after only a small amount of the metadata required to define the animation has actually arrived.

8 Saving Files

You may save images back out of "*kdu_show*" via the File menu options. This is of interest particularly if you have edited an image's metadata, or if the image has been obtained from a remote JPIP server – in that case, the saved file will contain whatever information has been received from the server up to the point when it is saved.

You may edit and resave a file over the top of the file you already have open. In this case, however, the file is actually saved under a different name, formed by appending the "~" character to the name of the currently open file, rather than overwriting it. When you close the open file, any such saved copy is automatically renamed so as to replace the one you had open. This happens even if you close the application (by any of the usual means, or by hitting the "q" accelerator). The File menu provides a convenient "Save and Reload" option, which combines the above operations of saving over the current file and then re-opening it.

You can save files as raw codestreams (choose a suffix of ".j2c" or ".j2k"), simple JP2 files (choose a suffix of ".jp2") or JPX files (choose a suffix of ".jpx" or ".jpf"). If you are currently viewing a motion JPEG2000 file (MJ2), you can save current frame as a JP2 file or a raw codestream, but you cannot currently save the entire movie.

The File menu provides two additional variations on the "Save As" option, which enable you to force the use of external links to the codestreams or force the embedding of codestreams in the saved file. In most cases, all relevant codestreams are already embedded in the images you open with "*kdu_show*" and the "Save As" option retains this embedding. However, JPX files can contain links to codestreams which may be found in other files; the "Save As" option preserves such links rather than embedding the codestreams. "Save As Linked" forces the use of links, meaning that codestreams which are currently embedded in the file you have open will be recorded as links into that file. Linked codestreams are very useful while you are editing metadata for a very large image or collection of images, since each time you save your results, only the metadata structure need actually be written to disk. Of course, with linked codestreams you have to be careful not to delete or overwrite the file which contains the linked codestreams. "*kdu_show*" does its best to keep track of links into any of the files it has open and prevent you from saving over any of those files, but it can't be aware of files it does not have open, so you could really get into a mess if you manually delete or move content around.