

Non-Destructive Imaging: An Evolution of Rendering Technology

by Peter Krogh

TABLE OF CONTENTS

1	Introduction
1	Part I—The evolution of non-destructive imaging
5	Part II—The rendering engine
7	Rendering types
10	Previews
12	Part III—Catalog-based non-destructive imaging
14	File browsers versus cataloging software
15	The advantages of catalog-based PIE software
17	Conclusion
18	Glossary

SO WHAT IS DESTRUCTIVE IMAGING?

You might be tempted to ask, why destroy your photos? Aren't you supposed to make them *better*? The word destructive is not referring to destruction of the photo's goodness, but rather destruction of the original data in the image.

So while you may make a photo "better" by lightening up shadow detail, you may do it in a way that destroys the original information. This paper concentrates on ways to adjust photos without losing the original information.

Introduction

Over the last couple of decades, the term *non-destructive* has been applied to many different kinds of imaging technologies. While the term is useful as a broad classification, it covers so much ground that it can often add more confusion than clarity, particularly when deciding which type of non-destructive imaging offers the best tools for the task at hand. This confusion is exacerbated by the fact that there are no accepted terms to describe the different kinds of *non-destructive imaging* (NDI) or the components of the various methods.

In general, NDI refers to imaging processes where the *source image* may be adjusted in a way that leaves the original data intact. There are several different ways to accomplish this type of adjustment, each with its own uses, strengths, and weaknesses.

In this paper, we'll look at several different NDI technologies and offer a nomenclature to distinguish among the methods. We'll put a name to the processes, so that the user can more clearly understand the components, and explain how the user can benefit from each process. We'll also describe the relationships between the components and provide a visual representation of those relationships.

We'll take an extended look at a new kind of NDI software—one that is based around the concept of a library, or a *catalog*. We'll describe its structure, uses, advantages, and drawbacks.

Part I—The evolution of non-destructive imaging

Let's take a short trip through the history of non-destructive imaging and see how we got to where we are today. We'll look at the development of non-destructive imaging technologies and examine how they have changed.

Overview

Non-destructive editing technologies are not new, nor are they confined to imaging. We have had the ability to preserve multiple versions of individual photos since imaging software first arrived, and therefore we have always had some sort of NDI. Additionally, software for graphic design and video editing has long had non-destructive capabilities, through the use of *referenced files*. So what's the big deal?

While NDI has been around for a long while, digital photography changes the landscape in some very important ways. Because many raw camera formats are proprietary, most imaging software cannot write changes back to the original files. This has spurred the development of imaging software that works with referenced files, rather than by changing the original source image. What started as a handicap—the inability to alter the original file—has led to great improvements in efficiency of workflow, storage requirements, and image management.

The efficiencies this new class of software brings are quite important for the challenges presented by digital photography. Some of those challenges include:

- The creation of a vast number of photos
- The need to apply identical adjustments to many photos in one operation
- The need to interpret a single source image in multiple ways (black and white, and color, for instance)

- The desire to take advantage of rapidly evolving imaging technology to reinterpret images with more capable software in the future

Additionally, the tools we can use to manage photos work very well with the rendering tools in NDI software. The *rendering metadata* and the *descriptive metadata* can be managed by a single application, offering the user a single environment to describe, group, adjust, manage, and output images.

Film photography—originals, prints, and duplicates

As we think about digital photography, it will be helpful at times to compare it to the world of film photography. With the exception of a few specialized retouching processes, the *original* film image, once created, never changes. This singular and unchanging original may be archived in a notebook and can be used as a source photo for multiple interpretations, usually in the form of prints. Film copies of the photo—generally called *duplicates*—are not identical copies, due to the inherent limitations of the analog film copying process. Furthermore, film duplicates are often created with corrections for cropping or color balance. In many ways, film duplicates should be thought of as prints of the image—further interpretations of the image rendered onto film instead of paper.

In digital photography, we also have a single original—the image data produced by the camera. And a print on paper is still a print. But the term duplicates does not offer enough nuance to be all-inclusive. You might have an exact duplicate of the photo file—a copy—or you might have a version of the photo file that has had some adjustments incorporated—which we will call a *derivative file*, since it's derived from the original.

Much of what we'll look at in this paper centers around the concepts of original and derivative files—how they are stored, adjusted, managed, and made use of.

Derivative file NDI—Save As

NDI has always been achievable by simply saving the file as a new file, once adjustments were made. By preserving the original file in its original state, the user is free to make additional derivative files without compromising the integrity of the source image.

Early users of Adobe® Photoshop® would frequently use the Save As command to create new files when changes were made, to ensure that they could go back to a prior version. These earlier versions might be needed if errors were made during the imaging process or if the user wanted to explore a different interpretation of the photo.

While this approach enabled the user to save both the original and the adjusted version, it came at a high price. The creation of multiple variations of a photo could bring great confusion. Often users would append a term to the filename to indicate its adjustment, but that might be hard to sort out later. Trying to sort out the difference between `Picture1_final3_Sharpen.TIFF` and `Picture1_final3_realfinal.TIFF` would often not be worth the effort.

Use of derivative files to enable NDI also brings a high price tag in terms of storage. Since the entire file needs to be resaved, even small changes require resources to duplicate and archive the entire file. So while use of derivatives enables NDI, it does not do it very efficiently.

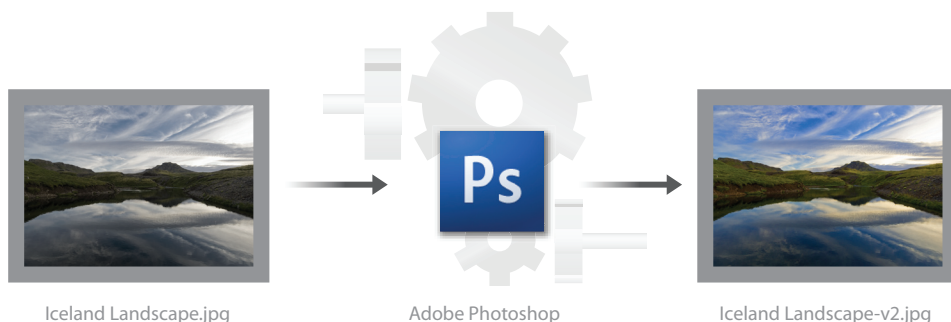


Figure 1: Non-destructive imaging has always been possible by creating derivative files through the use of the Save As command.

Self-referenced NDI—layers and more layers

In Photoshop 3, layers were introduced and it became possible to save multiple versions of the same image within a single file. This helped with the management difficulties associated with derivative file workflow but did nothing to address the storage needs. In Photoshop 4, adjustment layers were introduced, and it became possible to wrap up the source image with a set of instructions (or many sets of instructions) for rendering the photo. Let's call this *self-referenced NDI*.

In order to complete the self-referenced package, the file needs to be built with a *composite layer* or *embedded preview* on top. This component of the file structure shows what the image looks like when all the adjustments are applied. If the file is built without the preview, then it can only be correctly displayed when it's sent through a rendering engine that correctly recreates the modifications specified by the non-destructive layers. (More on this later.)

Using self-referencing files offers some significant advantages over derivative file workflow. Since the instructions to adjust the image take up much less space than creating an additional file, it's much more economical in terms of storage. And since you can name adjustment layers for an effect—as well as clearly see what the adjustment does—it becomes much easier to sort out what has been done to an image.

Using adjustment layers also offers you the ability to create a separate interpretation of the file—a black-and-white version, for instance—while still making use of some of the work done for other versions, such as retouching.

As the capability of NDI through adjustment layers has grown, it has become possible to keep access to a virtually infinite number of variations of a particular file inside a single TIFF or Photoshop Document (PSD) file. While this is great for *master file* workflow, it does not work so well for the kinds of tasks we need to do with a digital photography collection.

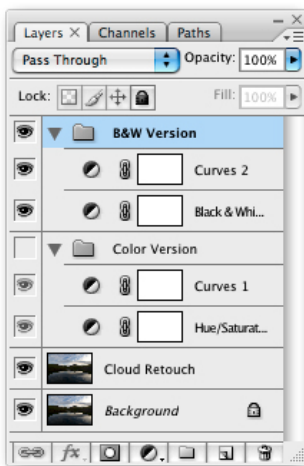


Figure 2: Adjustment layers offer the ability to apply non-destructive changes to image files. In this case, both the black-and-white and color versions can take advantage of the retouching done on a lower layer.

LAYERS AND LAYER SETS

By using layers and layer sets effectively, many different versions of an image can be stored in a single file. While different versions can be stored in a single document, only one version will be visible at one time. This composite is what you see in Photoshop and is used to build the preview at the time the file is saved.

The preview is essential for other software to see the effects of the Photoshop adjustment layers. A built-in preview is standard in TIFF files but is a selectable option in PSD files.

Pure instruction sets—video editing, page design, and Live Picture

Early on, some software programs moved to a structure that assumed that the original source would not be altered. Instead, control was achieved by means of saved instructions that could be applied to the work once editing was finished and output was needed. This has been common practice for all graphic design and video-editing software for some time. A photo-editing application, Live Picture, also made use of this structure for still photos, by use of *proxy images*.

In design and video programs, the use of referenced files was adopted to get around hardware and performance limitations. Because a magazine layout file grew too large when the photos were embedded, software was developed that used a small proxy of the photo and referenced the high-resolution file when output was needed. In video editing, referenced file editing is again necessary because file sizes are too large and processors are too slow to work with the full-resolution assets.

We're going to call the editing of images by creating instructions or parameters *parametric image editing (PIE)*. For the purposes of this paper, we'll call the software that does this *PIE software*.

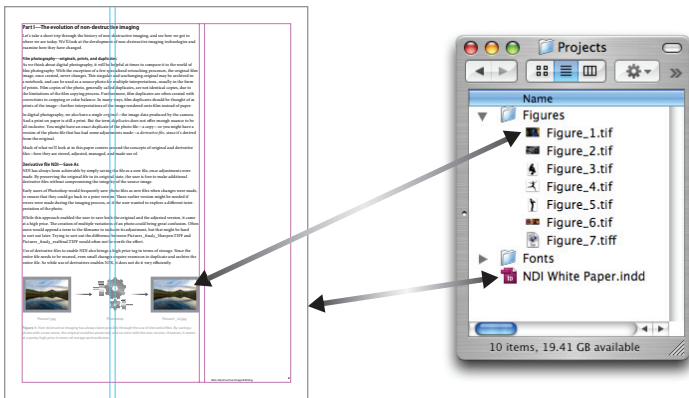


Figure 3: Graphic design software, such as InDesign®, works by means of proxy images. The layout software creates a low-resolution version of the file to use in the design process. It references a high-resolution version of the file that's used for output. This structure offers the ability to keep documents relatively small and preserves the creative flexibility to make changes to all elements of the project.

Live Picture was an image-editing program that used a small version of the file—a proxy—to show the user the effects of any adjustment tools. By working on small files, adjustment decisions could be made quickly. When a final, derivative version of the image was needed, the original source image could be sent through the Live Picture *rendering engine* for the creation of high-resolution output.

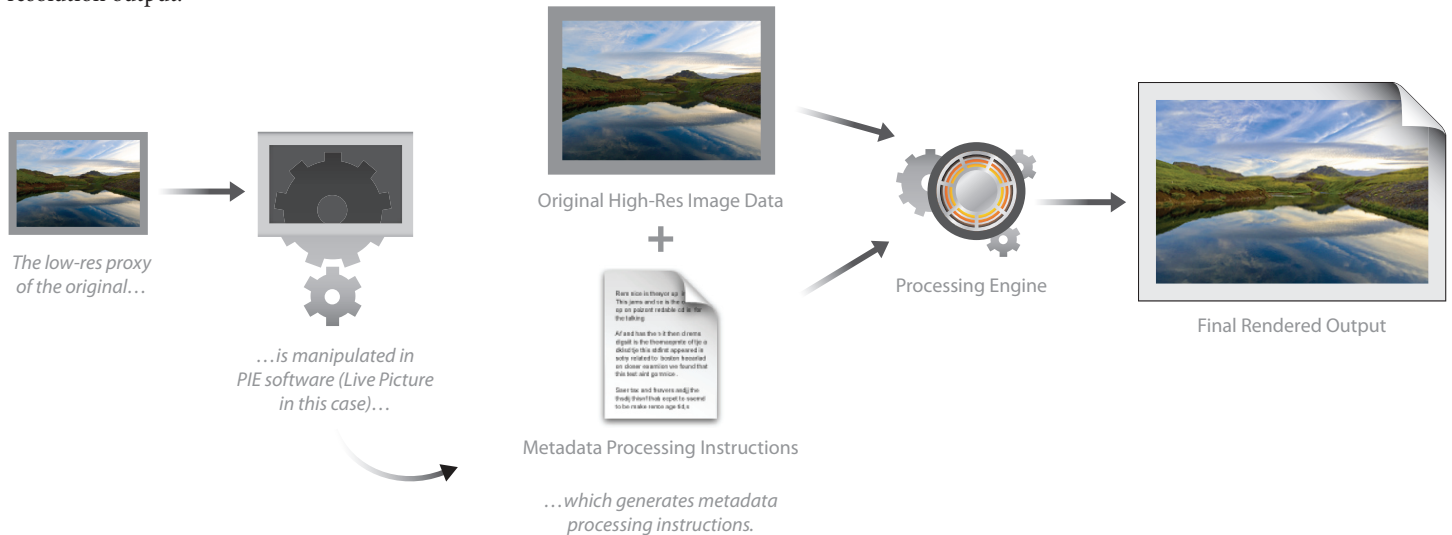


Figure 4: Live Picture used a small proxy of the image file to show the effect of changes to the photo. When the user was satisfied with the image, the same adjustments were applied to a larger version of the file.

On the accounting side, working with proxy images and referenced files offers performance advantages and great savings on data storage. It also offers something for the creative side—the ability to experiment with alternate versions and to refine your work easily. Because the saved work is just a set of instructions—and instructions are easy to duplicate and refine—you can freely experiment with branching the work off from any point in the creative process, without fear of losing good work when in search of better.

Digital camera raw files—parametric image editing takes off

Although video and design software has been using referenced files for many years, it was not until the digital camera became popular that referenced imaging took off as the method for adjusting still photos. To some degree, referenced imaging was making lemonade out of lemons. At the same time, it was just the right combination of user need, technological development, and maturing software.

Nearly all digital cameras produce a *mosaiced image* as a *raw file*. Although this may look like a regular photo once it's displayed in software, in reality it is a checkerboard of red, green, and blue values that are transformed into an RGB image when the file is processed by a *Raw File Converter*. When we look at one of these raw photos on our computer screen, we either see a *live rendering* of the mosaiced data, or we see a *fixed rendering* in the form of a JPEG preview that has been made by the camera and placed inside the raw file.

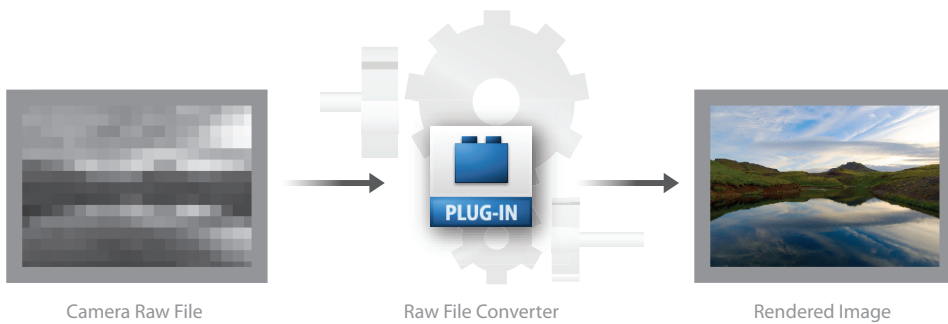


Figure 5: Raw files from digital cameras record the information in a mosaiced pattern that must be converted to RGB color to be displayed. This mosaiced pattern is a useful format for camera data—because the sensor records it that way—but it is not an appropriate format to store rendered images. Therefore, working with camera raw files will always be a one-way street, as far as the source image data is concerned. Source data can be read from but not written to, which forces raw software to non-destructively interpret the file rather than altering it.

There's a critical issue that prevents desktop applications from resaving raw files. With few exceptions (for example, those cameras that shoot a Digital Negative [DNG] raw format in camera), raw images out of a digital camera are proprietary and undocumented. This means that third-party software should generally not make any alterations to a raw file because of the possibility of corrupting the file.

These two characteristics—the encoded color information and the undocumented file structure—have led to the development of photo-editing software that works only as NDI software and that will never change the source image. While this was a painful transition for early adopters of digital cameras, it has led to wonderful developments in photo software and workflow. These include multiple photo manipulation, unlimited undo, and—thanks to just-in-time imaging software—great space savings.

Some of the top benefits of NDI include:

- **Multiple photo manipulation:** Digital cameras have the ability to produce large numbers of similar files in a way that scanning photos never would have. Parametric image editing (PIE)—the process of editing images by creating instructions or parameters for adjustment—makes it easy to apply settings made for one photo to many others very quickly.
- **Unlimited undo:** Since all image adjustments are saved simply as processing instructions, it's easy to change those instructions to create a different interpretation of an image. In order to save the additional variations, all the user needs to do is save the instructions that are used to create the variations.
- **Space savings:** Now that computers are fast enough to create renderings from the images as needed—let's call this *just-in-time rendering*—the user can build a library of images that consists largely of the source images and the rendering instructions. Since instructions are a lot smaller than pixels to store, the storage requirements are greatly reduced.

Part II—The rendering engine

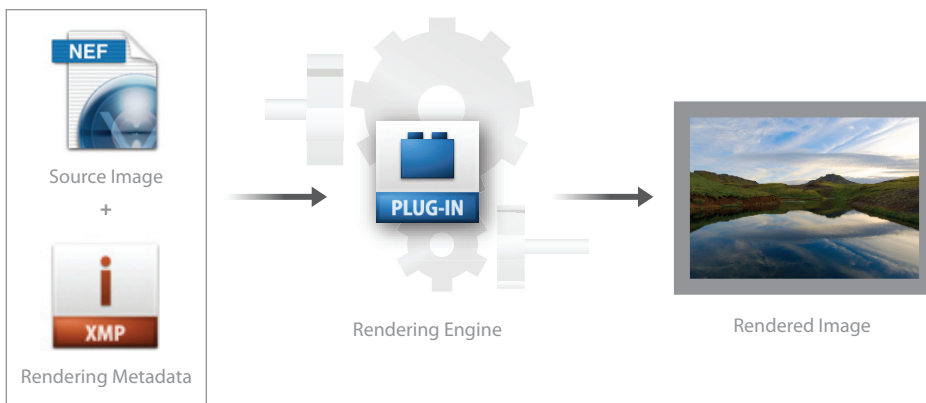


Figure 6: The rendering engine in its simplest form: The source image and your processing settings (in the form of rendering metadata) are processed in the rendering software and a finished image is created.

Understanding rendering

As you start working with referenced-image NDI software, it's important to understand the rendering engine. There are some important differences in how referenced images work and in how *fixed-rendered* images work. Let's examine this.

Fixed rendering refers to any image that has its rendering described in pixel information, such as an RGB or a grayscale value. The fixed rendering is not dependent on any particular rendering engine to display according to the user's wishes. In many ways, we can think of fixed rendering as a print of the image, because, like a print, the user has made a fixed interpretation of the original image that is not dependent on the presence of the original negative or the enlarger.

When the RGB value is attached to an International Color Consortium (ICC) profile (which gives the RGB number an objective meaning), each distinct three-number combination describes an exact color. The end result is a single correct way to display that pixel's color, within the limitations of the viewing device (no monitor can display all colors).

A raw file, however, does not have the same objective color value that a fixed-rendered image does. The color information in the raw file needs to be sent through a rendering engine to be turned into standard RGB color. Each rendering engine will decode the color information in a different way. Additionally, each rendering engine has controls to adjust how the raw data is decoded and displayed.

This means that the "color" of a particular pixel in a raw image is a combination of three things: the original image data, the rendering engine's mathematical formulas, and the settings applied by the user to the rendering software. While we can't change the original image data, we can decide to use a different rendering engine (a different program to convert the raw files), and we can decide to change the settings used to interpret the image.

It's important to understand that there is no objectively correct way to decode and interpret the raw data. There is only subjective interpretation, based on the mathematical formulas that are built into the rendering engine and the user's settings. In order to make a version permanent and available to all different programs, one needs to create a fixed-rendered version of the file, so that the color values are exactly specified and not dependent on a particular rendering engine.

Just-in-time rendering versus caching

Software that works with a rendering engine can be thought of as just-in-time rendering software. This term, from the manufacturing industry, refers to a setup where work is completed and delivered only as necessary, and not before. Since the source images can be interpreted in many different ways, it does not make sense to spend a lot of resources prerendering the images. It's quite likely, for instance, that as you are getting ready to send photos to a printer, you may want to perform a few last-minute adjustments to the images.

In a just-in-time environment, the software will perform the rendering as needed for the task at hand. If the user needs to print, the current version of the photo will be rendered in an appropriate way, and that information will be sent to the printer. If the user wants to send it to another program, like Photoshop, for pixel editing, a rendered 16-bit version of the file can be created and saved to disk once it's needed, but not before.

Just-in-time rendering saves lots of disk space and saves the user a lot of work, but that does not mean that every rendering task is saved for the last second. The need to look through lots of images requires that some rendering be done up front, so that the rendering engine does not fall behind. Previews of the images are created and cached, so that images can be examined quickly.

The cached version of the images are simply proxies of the larger source files. They are often created as JPEG files that compress the color and detail of the photo to some degree. They may be full size, or they may be smaller in dimension than the original files. In Part III of this paper, we'll look at some differences in how the cached versions of the images are handled.

FOR MORE INFORMATION

This paper describes a number of aspects of the rendering engine, particularly as they relate to non-destructive imaging. For a more thorough discussion of the issues related to rendering and digital camera raw files, please refer to the Adobe technical paper "Rendering the Print: The Art of Digital Photography" by Karl Lang.

Each rendering engine is different

One of the most common points of confusion with raw file rendering is the difference between rendering engines. Photos that looked one way on the back of the camera or when opened with one raw file converter may look quite different when opened with another piece of software. It's important to understand that your camera has a raw file converter and a rendering engine of its own, and that the rendering engine on the camera will be different from the engine in any third-party desktop software.

When you shoot a picture on a digital camera, the raw file converter in the camera creates a JPEG file from the raw file and places the JPEG image inside the raw file as a preview. This is the picture that shows up on the camera's LCD screen and in some third-party browsing software like Photo Mechanic. While this is the first interpretation of the image, it's not necessarily any more accurate than the one generated by third-party software. The engineers who built the camera (and its raw file conversion engine) are—like third-party developers—trying to make the most pleasing rendition and not necessarily the most accurate one.

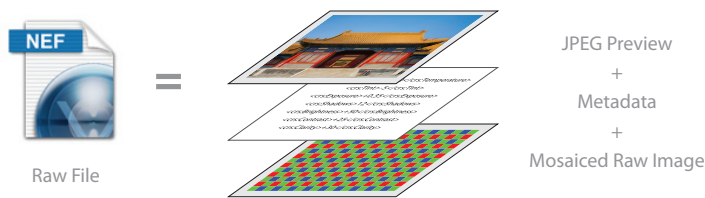


Figure 7: A raw image produced from a digital camera contains more than you might think. Not only is there the original raw image data, but there is metadata created by the camera, including the user settings and time the photo was shot. Additionally, there is a preview of the image file, as converted by the camera's onboard raw file converter. This JPEG image is what is displayed on the back of the camera. Since this image goes through the proprietary converter in the camera, it will not exactly match the rendering produced by third-party raw converters.

Rendering types

Because PIE software only shows you an interpretation of the photo, it's important to understand something about how that interpretation is created, stored, and exported. To do this we need to give names to various kinds of interpretations. Let's break this out into several different classes of images.

Source image

This is the starting point in the imaging system—the original image. While we have mostly been talking about raw images, PIE software is now being used on source images that come in to the program as fully processed RGB images, such as JPEGs or TIFFs. Many users find they want to extend the benefits of parametric image editing software to these previously rendered images.

While we're talking about sources, let's make a distinction between the *source image* and the *source file*. As described earlier, many image file types contain several versions of an image inside the same file, such as the raw data and a JPEG preview. The source image is the image data itself, while the source file is the document that the image data has been saved into.

Engine-dependent rendering

Let's use the term *engine-dependent rendering* to refer to any system where the user's interpretation of the file can only be seen by sending the image through a particular rendering engine. This is distinct from a fixed rendering, such as a JPEG adjusted with Photoshop or Google Picasa™, where the finished photo can be displayed identically by any color-managed imaging program. When you look at an engine-dependent rendering, you might be looking at a live rendering (like the Develop module of Adobe® Photoshop® Lightroom™), or you might be looking at a preview that has been cached to disc, as in the Library module of Photoshop Lightroom.

Live rendering

A live rendering is a view of the image that only exists when the source image is loaded into the software. In the darkroom, this would be akin to putting a slide in the enlarger and projecting it on the baseboard. You can see the photo, but the resulting image is no more permanent, nor easily shared, than the image coming out of the enlarger lens. (We'll discuss previews—cached live renderings—shortly.) This rendering can be considered more/less as a temporary preview.

Default rendering

The first time you look at an image in a parametric image editor, you see the default rendering of that image. It's important to understand that this is not a "correct" or "un-retouched" version of the image. The default rendering is just a starting point—one that is determined by how the software is set up. It's very much like putting a slide in an enlarger and turning the lamp on. It may be a good way to interpret the image, or it may need lots of adjustment to please you. Most PIE software will let you create your own defaults— to be used as a default for a specific camera, for instance—that can be used as your own starting points for image adjustment. This may include such items as turning auto corrections on or off, or other settings.

Saved engine-dependent rendering

Once you have made basic adjustments and color corrections to a photo, you may wish to save the settings you used so that you can access this version in the future. Some PIE software will automatically store the last settings that you used when adjusting an image, and some will also cache the preview that was generated in the process. Using PIE software, you can now generate an unlimited number of different renderings of any given photo. Storing and recalling multiple renderings for a single photo is discussed in Part III—Catalog-based non-destructive imaging.

To use our enlarger metaphor, saving the rendering settings would be similar to writing down your enlarger settings once you have obtained a print you like. With a complete recording of the settings, you can reload the image and reset the controls in order to reproduce that particular rendering of the image.

Rendering metadata

When you save a particular rendering, you do so by saving the *rendering metadata*. This metadata includes the set of instructions that the rendering engine uses to process the image. Most raw files come with some metadata that is used during the rendering process, such as the white balance setting that the user selected in camera. (In general, most of the in-camera rendering instructions that are saved with the photo at the time of exposure are only useful to the camera manufacturer's software.)

In fact, the rendering metadata created by any PIE software will only be useful to that particular rendering engine. Adobe® Photoshop® Camera Raw and Photoshop Lightroom, for instance, both use the same rendering engine when interpreting images, so they will give identical renderings when the rendering metadata is identical.

Rendering metadata for many programs is saved as XMP data. An example of XMP information is shown below. These settings can live inside the file (depending on the format), in a sidecar file, or in a database, just like the informational metadata that describes what the image is about. We will discuss this further in the next section, Fixed renderings—derivatives and prints.

```
<rdf:Description rdf:about="" xmlns:crs="http://ns.adobe.com/camera-raw-settings/1.0/">
<crs:WhiteBalance>As Shot</crs:WhiteBalance>
<crs:Temperature>4450</crs:Temperature>
<crs:Tint>-5</crs:Tint>
<crs:Exposure>+0.35</crs:Exposure>
<crs:Shadows>12</crs:Shadows>
<crs:Brightness>+50</crs:Brightness>
<crs:Contrast>+25</crs:Contrast>
</rdf:Description>
```

Figure 8: Here's a bit of the rendering metadata that came out of a sidecar file created by Adobe Photoshop Camera Raw. The information at the start of the metadata tells the program that this is information for Adobe Photoshop Camera Raw (or Photoshop Lightroom) to use when interpreting an image. It's followed by the specific settings for the controls in the engine. As long as this information stays with the file, the program that created it can re-create the rendering that it describes.

Fixed renderings—derivatives and prints

In the course of working with image files, there are many times when it's useful to create a fixed rendering of one sort or another. Perhaps it's needed so that the photo can be easily identified and examined; sometimes it will be desired because a photo is "finished" and needs to be sent off to another person or software application. Sometimes that fixed rendering will be made in the form of a print. Let's take a look at how and when fixed renderings can be made and where they can live.

Baking in the settings—derivative files

When you use PIE software to create a new image file according to the user's adjustments, you've made a *derivative file*. This new file, derived from the original, is an output-referred version of the original photo and therefore does not need the raw processing engine to be displayed according to the user's adjustments.

Derivative files are useful for a lot of tasks, like building web galleries, sending images to printers, proofing, and opening images in other programs. When you make the derivative files you are "baking in" the settings. And once they're cooked into the pixels, they can't be uncooked. That is, you've decided on these settings and can't undo them. A derivative file is very much like a print. You make it when you are satisfied with the adjustment you have made to the photo. Once it's been created, you no longer need the enlarger—it's a freestanding image. And just like making a print, it takes up storage resources if you want to keep it.

Prints

A print is also a fixed rendering. As the highest-cost way of making a fixed rendering—in terms of time, monetary cost, and storage needs—it generally only makes sense to create a print when there is an actual need for one.

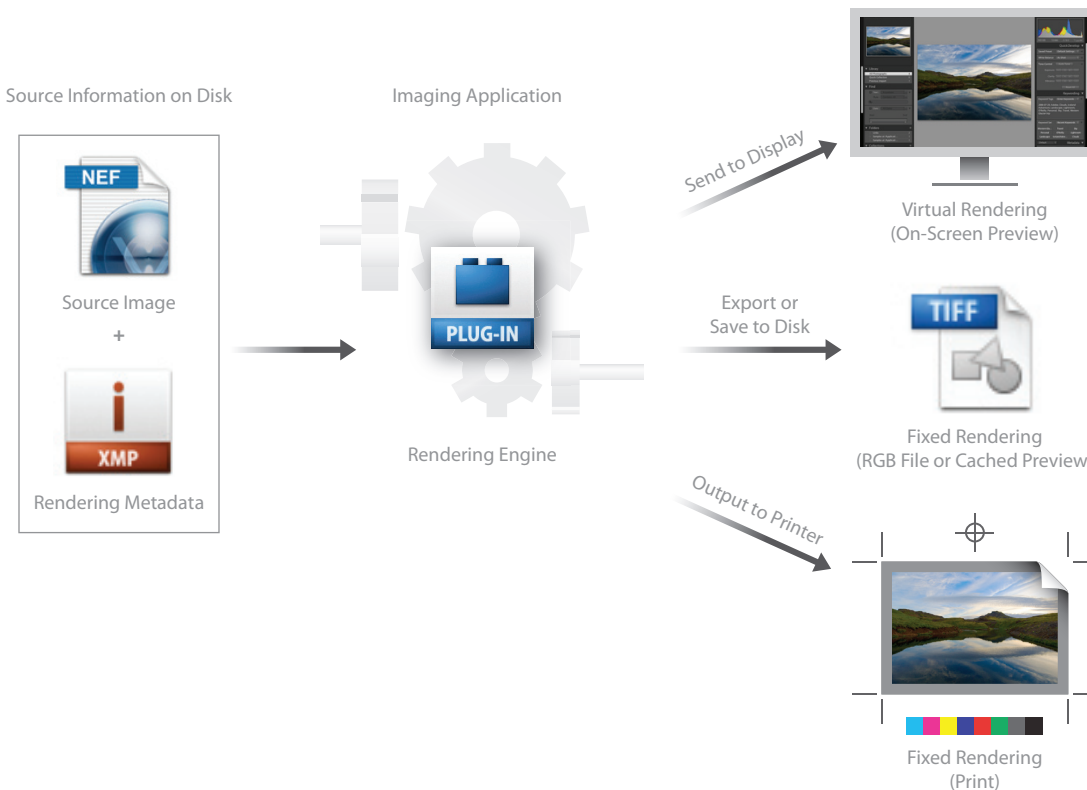


Figure 9: Now that we have the components described, let's look at how they work together. The source image and the rendering metadata are sent through the rendering engine. When you look at the photo that comes out the other end in a program like Adobe® Bridge or Photoshop Lightroom, you are only seeing a temporary rendering of it. If you wish to save a stand-alone version of the adjusted photo, you need to create a whole new file, since the programs don't change the source data. You can also create a whole new copy of the photo as a print.

Previews

One of the most useful tools in non-destructive imaging is the *preview*—usually a JPEG version of the image that lives in the image file, alongside the image file, or in a catalog of image files. A preview is a semipermanent fixed rendering of the image that shows the adjustments made by the rendering engine.

Previews are created at many stages of the rendering process to speed up the display of the images or to create a version that can be displayed identically by any program. A preview is a proxy of the image, useful in selecting, managing, and sometimes outputting photos.

Previews that are created by a rendering engine are dependent upon that particular engine. As such, they are only one interpretation of the file. Some programs will offer the user control over when, where, how large, and with what settings a preview is created. Some previews are created automatically with little or no user control.

Let's look at where previews show up and how they are used.

Embedded previews—raw files

Raw files created by digital cameras must have a preview inside the raw file itself in order to be displayed as a normal full-color image. Most current digital cameras will put a preview inside the raw file that has the same resolution as the raw data itself. (As a matter of fact, there are usually several previews of differing resolutions inside each raw file.) This preview can be accessed and properly displayed even by software or operating systems that cannot decode the raw file data.

Previews in raw files are built according to the user's settings for the camera, such as white balance, sharpening, and contrast control. Previews in raw files can only be created by software made by the camera manufacturer, unless the raw file is in Digital Negative (DNG) format, which supports an open standard for writing and re-writing the preview by third-party software.

Even if a camera does not create a DNG in camera, raw files can be converted to DNG at a later date. These DNG conversions retain the flexibility of adjustment that the proprietary raw files had but offer a number of additional capabilities. The ability for any third-party raw processing engine to embed its own engine-dependent rendering (EDR) preview and rendering settings into the DNG file is one of the most compelling reasons to consider the format for archival storage. TIFF and JPEG files can also be converted to DNG after editing in order to wrap the source image up with a rendered preview of the adjusted photo.

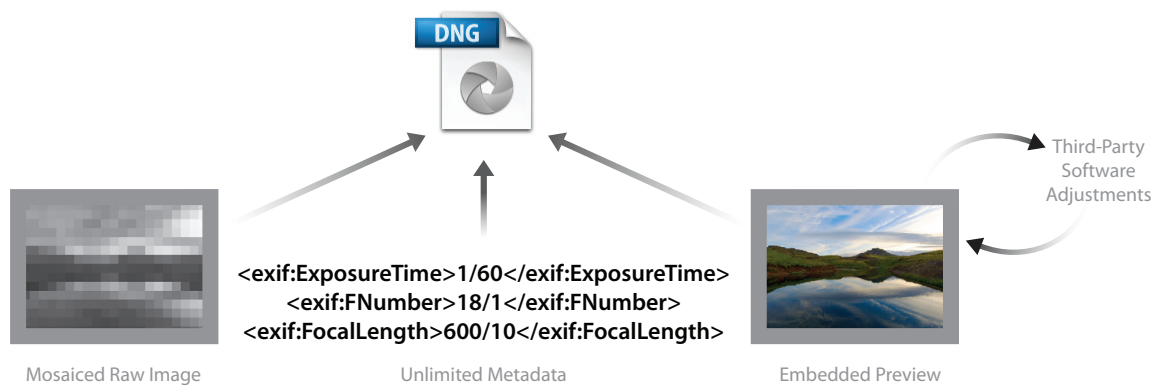


Figure 10: Proprietary raw files, such as CR2 and NEF, contain a preview that can only be modified by manufacturer's software. Because DNG is an openly documented file format, any software can replace the preview with one that reflects the user's adjustments. This enables the user to bundle up the source image data, the metadata, and a semi-permanent fixed rendering into a single file.

Cached previews—file browsers

A file browser will often create its own previews to speed up the display of the photos. These previews will generally be saved as files that are hidden to the user, so that they are not confused with versions of the file that the user might have created as derivative files. Adobe Bridge creates previews that live in a central database in the Users directory and can also save the previews to an invisible file inside the folder in which the images are stored (invisible in Bridge CS3 only).

Many browsers, such as Photo Mechanic, will make extensive use of the preview embedded in the raw file to boost speed. Because they don't have their own rendering engines, they are dependent on the embedded preview to be able to display the image.

Cached previews—cataloging software

Cataloging software makes use of previews in much the same way browsing software does, but with added capabilities. The previews are used to speed up the display of the images, and in the case of cataloging software that does not include a rendering engine, embedded previews are used as a proxy for the adjusted image.

In addition, cataloging software also has the capability to work with off-line media—files that are not currently connected to the computer. So rather than storing the previews in an invisible way like a browser, cataloging software needs to include the previews in the catalog itself. And because the size of the previews can grow very large (when the catalog contains a large number of photos), cataloging software will generally offer the user some control over how the previews are generated, saved, and transferred between computers.

Cataloging software that has its own rendering engine can make and remake previews according to the user's adjustments of the photos. It can also make use of the previews for certain kinds of output, like draft-quality printing. We'll look at this more closely in Part III.

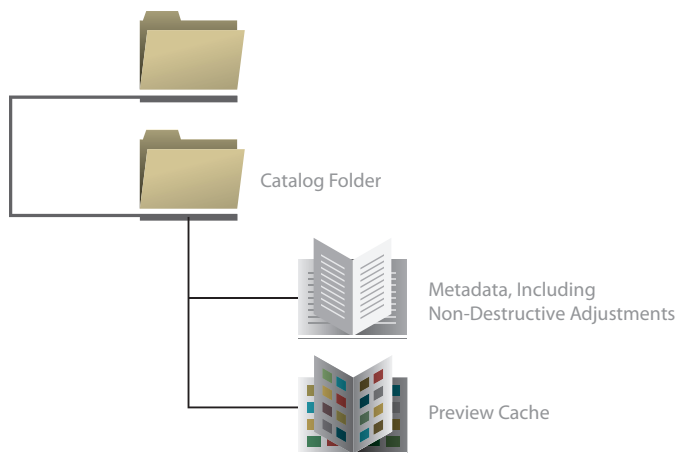
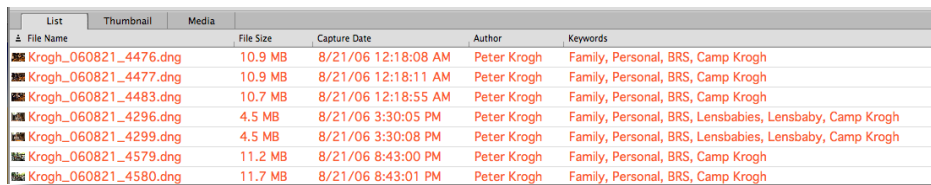


Figure 11: Cataloging software creates previews of the photos and stores them with the rest of the catalog metadata. These previews are used to identify and organize the photos. Some output created by the cataloging software, such as proof prints or web galleries, may use the preview as a proxy for the file in the interest of speed.

Part III—Catalog-based non-destructive imaging

While imaging applications were moving from destructive pixel pushing to referenced file NDI, another development was happening in the imaging arena. Cataloging programs, which were formerly geared to graphic designers, started to pay attention to the digital photography space. Rather than handling photos individually, these programs use a library model for their structure.

In this model a catalog holds all the information about many photo files. The files are stored in a single folder, in multiple folders, or even across multiple hard drives. The software knows about each photo, and can control it from a single location. Because the software has collected all the metadata about the photos in a central location—and because the software can create new metadata as the user deems necessary—the catalog becomes a versatile place to know about entire collections, and to control and manage the photos.



List	Thumbnail	Media			
File Name	File Size	Capture Date	Author	Keywords	
Krogh_060821_4476.dng	10.9 MB	8/21/06 12:18:08 AM	Peter Krogh	Family, Personal, BRS, Camp Krogh	
Krogh_060821_4477.dng	10.9 MB	8/21/06 12:18:11 AM	Peter Krogh	Family, Personal, BRS, Camp Krogh	
Krogh_060821_4483.dng	10.7 MB	8/21/06 12:18:55 AM	Peter Krogh	Family, Personal, BRS, Camp Krogh	
Krogh_060821_4296.dng	4.5 MB	8/21/06 3:30:05 PM	Peter Krogh	Family, Personal, BRS, Lensbabies, Lensbaby, Camp Krogh	
Krogh_060821_4299.dng	4.5 MB	8/21/06 3:30:08 PM	Peter Krogh	Family, Personal, BRS, Lensbabies, Lensbaby, Camp Krogh	
Krogh_060821_4579.dng	11.2 MB	8/21/06 8:43:00 PM	Peter Krogh	Family, Personal, BRS, Camp Krogh	
Krogh_060821_4580.dng	11.7 MB	8/21/06 8:43:01 PM	Peter Krogh	Family, Personal, BRS, Camp Krogh	

Figure 12: Here's a screenshot from a cataloging program—iView MediaPro (now Microsoft® Expression® Media). Each photo indexed by the catalog is a separate record. This view shows you a small fraction of the information a cataloging program would know about the photos. There's a thumbnail preview, followed by information about the file—in this case we're looking at the file size and the creation date of the photo. This is followed by some annotation information, including the author's name and any keywords that are attached to the photo. Cataloging software offers the ability to collect information about your entire photographic archive in one place and to organize the photos according to many different criteria.

Cataloging systems rely on metadata to know about and control the photos. Since non-destructive imaging relies on metadata to control the rendering of the photo, it's a natural fit for cataloging software and non-destructive imaging to grow together. This is just what's happened over the last few years, with the appearance of programs like Apple Aperture™ and Photoshop Lightroom.

The advantages that cataloging software offers for the raw photographer are numerous. The photographer (or image librarian) can see and segment an entire library based on the content, quality, and usage of the photos. This is very powerful information to use when making decisions about how much effort to spend adjusting photos or even which ones deserve any adjustment at all. It allows the photographer to make decisions about image editing and picture grouping concurrently and recursively.

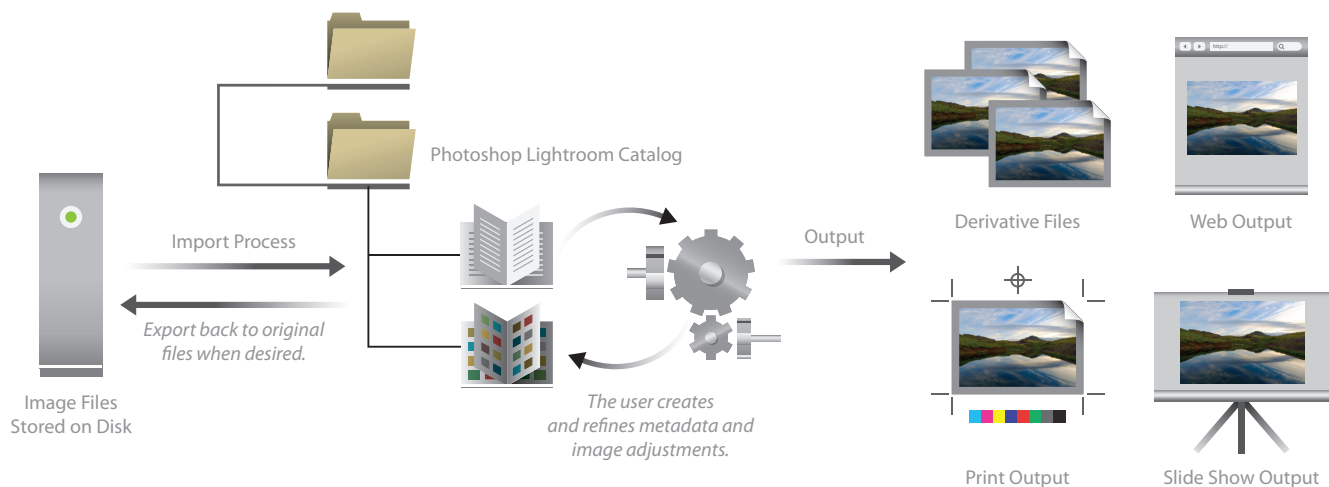


Figure 13: This figure describes how information flows in a catalog-based system. When photos are originally indexed, embedded metadata is collected in the catalog database. A preview of the file is created that enables the database to show what the photo looks like. When further work is performed on the collection, it is saved in the catalog. If the user wants to embed the information—either informational or adjustment metadata—back in the file, it needs to be exported back to the original files.

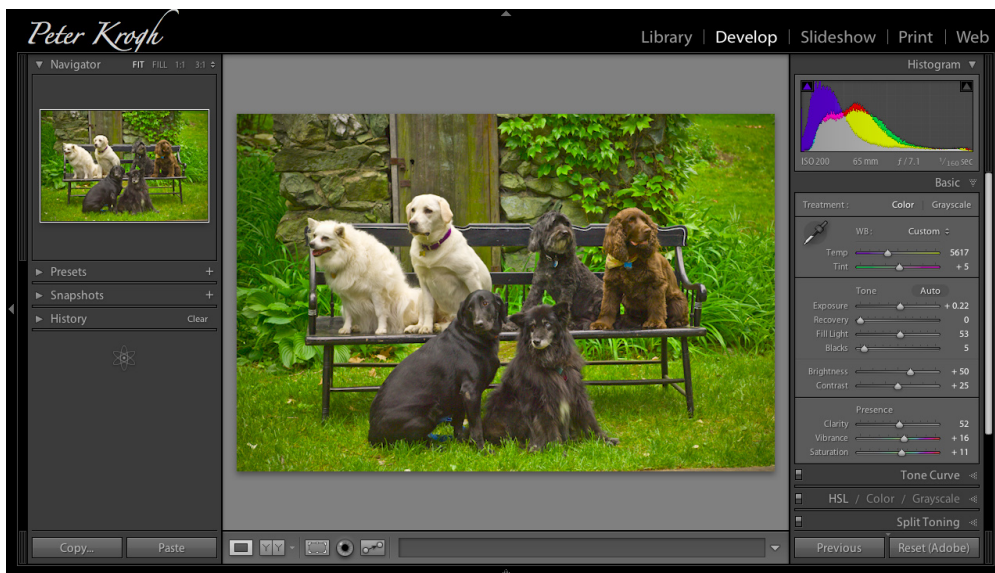
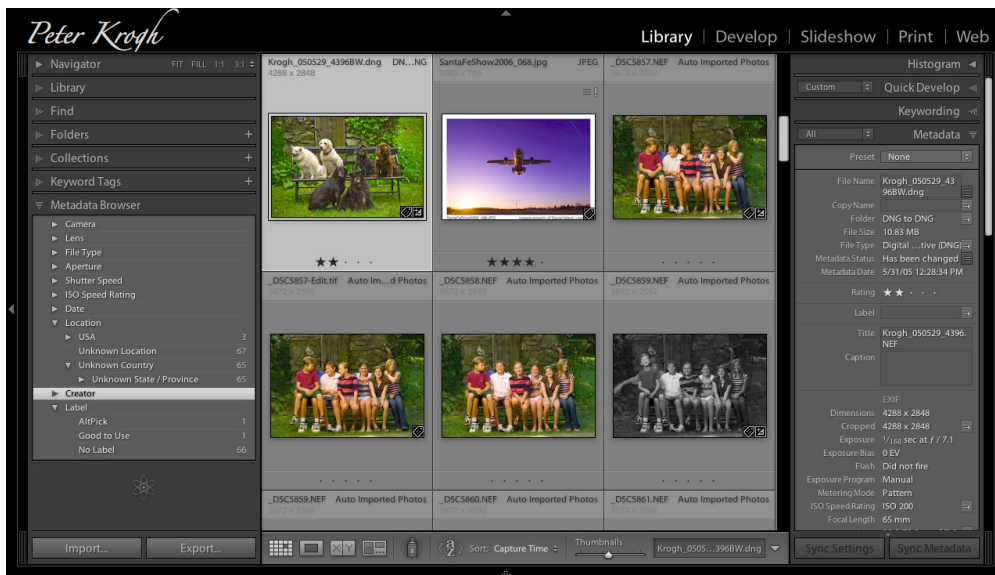


Figure 14: Photoshop Lightroom takes the functionality of a cataloging program and adds a rendering engine. Since both functions—organizing the photos and adjusting the interpretation of the photo—rely on metadata, it makes a very good match. By bringing these tools together, it's easier to combine picture selection with image editing.

The use of a cataloging application also enables the photographer to back up and protect photos in ways that a file browser directory-based workflow does not. Because the software knows what should be there, it can help you in the event of media failure or another significant problem. Furthermore, because the catalog knows about all the adjustments you have ever done to the photos, it can help you preserve that work in a very efficient manner.

Catalog-based non-destructive editing, however, is counterintuitive at first for many photographers. Until one understands how it works, moving between software programs can be mysterious and frustrating, and work might seem to disappear in the handoff. The balance of this paper will be devoted to providing a better understanding of the components of catalog-based NDI and how it works.

File browsers versus cataloging software

To the new user, the difference between browsing software and cataloging software may look more like an interface difference than anything else. Under the hood, however, there is a big difference between the programs.

The truth is in the file

While browsing software may have a database that it uses to speed up the actions, it never uses the database as a place to *save* information. File browsers merely *cache* information in their databases. That is, they stash it there for quick access but always look to the original source file to see if the cache is up to date and correct. If there is a mismatch between what is in the cache and what is in the file, the file wins, and the cache is updated automatically with the information from the file. This model is known as *the truth is in the file*.

When you make changes or add information to a photo with a file browser, the program will want to write those changes or information back to the source file as fast as possible. When you add keywords or make changes to the rendering metadata, the file browser is designed to offload that information back to the file (or its adjacent sidecar file), since that's the place where the truth lives. And if the file is no longer visible to the file browser—because it's been renamed, taken offline, or deleted—the file browser forgets the file ever existed.

This “truth is in the file” model is a very intuitive way to work, particularly if you are moving from the physical world to the digital world. If we think about our photo collection as a bunch of individual photos, it makes perfect sense that each freestanding image would contain everything we know about that picture.

The challenge of managing a digital photography collection looks a lot more like managing a forest than managing a bunch of trees, however. Once we factor in the challenges posed by working with large groups of photos, things seem different. And when we add the functionality made possible by database-driven catalogs, the advantages of the catalog model become very clear.

The truth is in the catalog

As distinct from the file browser model described above, with cataloging software the information about the photo is assumed to be most accurate in the catalog, rather than what may be residing in the file itself. Of course, much of the information came from the file originally and can be pushed back into the file when desired. But in the daily course of working with images, the most complete and current information about the image is assumed to be what's in the catalog database. We call this model *the truth is in the catalog*. Let's examine this a bit.

When photos are first indexed by the cataloging software, several things happen. A record is created for each photo file that will contain the information about it. A preview of the image is created and stored in a place that can be accessed by the catalog, even if the file is not currently available. Any metadata in the file gets harvested and is associated with the photo in the catalog. The metadata then gets parsed into groups, such as creator name, keywords, ratings, or locations.

When you add information to the image using the cataloging software, such as adding a keyword, you are just making a note in the catalog record for that photo. Unless you take some specific action, the original image file will remain untouched. (In Photoshop Lightroom, you can set a preference to write each change back to the file immediately—assuming that file is connected to the computer—but this can really slow down the program.) Likewise, when you make a rendering change to an image with catalog-based PIE software, you are changing the information in the catalog and in the preview that gets cached, but you are not necessarily touching the original file.

Because cataloging software has the capability to work with offline files, it *must* consider the catalog information to be the “truth.” Otherwise, as soon as the files are connected again, all the work done in the catalog would be lost, since it does not match what's in the files.

If you're used to working with browsers only, this can be puzzling at first. The cataloging software shows that a keyword has been placed “on” a photo, but an inspection of it with another program shows no keyword there. The keyword is associated with the photo in the catalog, but is not actually stored within the photo file. While this takes a bit of getting used to, it has a number of advantages, particularly for large and growing digital photo libraries.

Swapping the truth

At times, you'll want to take the work you have done in the catalog and make it available for other users or programs to see. For metadata this is most commonly done by writing the metadata back to the original file or its sidecar file. It gets a little stickier for the renderings.

Since the adjusted version of the photo is dependent on the rendering engine to display properly, pushing the settings into the source file only works if the image will be viewed with a program that has the identical rendering engine. There are a couple of options for making the adjustments visible to other programs.

If your library is made up of proprietary raw files (and now RGB files for Photoshop Lightroom and Adobe Photoshop Camera Raw 4), the only real option for making the rendering adjustments universally visible is to create derivative files. New JPEG, TIFF, PSD, or DNG files can show the adjusted image in a way that is accessible to nearly any program.

If your library is made up of DNG files, you can push an embedded preview back into the DNG file. This semipermanent fixed rendering can make the adjustments visible to any program and remains embedded in the source file itself for easy management. DNG is an ideal format for making the work done in catalog-based PIE software portable, since it's the only format that can contain the mosaiced source image, all the metadata, and a user-created fixed rendering.

The advantages of catalog-based PIE software

The catalog configuration leverages what computers do best and protects against some of their failings. Catalog-based PIE software can gather, sort, group, and work on photos in ways not otherwise possible. And just as important, it helps you to protect your images and your work from the kind of sudden and total loss that can happen to digital assets. Let's take a look at some of the benefits.

Associates information with one original

Catalog-based PIE software greatly simplifies the organizing and archiving of photos because so much work can be associated with a single original source image. Information can be created, stored, and cross-referenced in a single environment. Multiple renderings can be attached to a single source image and managed from that same environment.

One of the greatest sources of confusion and storage bloat is the saving of multiple versions of the same file. As discussed earlier, multiple versions are hard to sort and take up a lot of disk space. By building an image collection that refers back—as much as possible—to a single and unchanging source image, the user can achieve great savings in storage space, time, and confusion.

Creates useful groups from scattered images

Browser-based systems rely on folder structure as a central organizing principle (since they browse folders and drives). This makes it difficult to bring together images that may not be stored near each other. Ongoing projects and repeat clients generate work that is created at different times and thus may be widely scattered in the file system. Since cataloging programs organize by metadata, proximity in the directory structure is largely irrelevant.

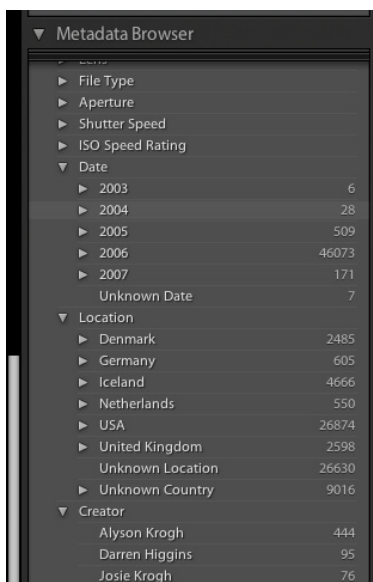


Figure 15: The Metadata Browser in Photoshop Lightroom lets a user look at an entire catalog according to many different criteria. This screenshot shows the kinds of metadata that Photoshop Lightroom harvests and gathers together—date created, location, and creator.

By gathering this information in a single place, the user can quickly find particular images that relate to each other. And by cross-referencing the metadata—find all images by Alyson, in North Carolina, created in August 2006—it's easy to find sub-groups of pictures, even if they are nowhere near each other in the file system.

Works with offline images

One of the most immediate benefits of cataloging software is that you can have access to photos that may not be currently connected to your computer. If you travel, and your collection is too big to bring, or if you have some photos archived offline, then cataloging software lets you look through, tag, and group your photos as if they were present.

Eases backup and restoration tasks

One of the most important needs in any storage system is the ability to efficiently and securely back up data, and to be able to restore that data in the event of some kind of media failure. The cataloging software model brings a simplicity to the process that can't be matched with any other imaging system.

Because the original image is never changed by the software, it can be archived and backed up early in the process, and does not need further backup, even after refinement of the metadata or image adjustments. All the work done to the images in the cataloging software is represented by the metadata in the catalog. Because the metadata is a small fraction of the size of the image data, it's a much more manageable task to back up this ever-changing data separately, without having to worry about the unchanging image data.

In the event of drive failure, restoring a catalog-based system is a relatively straightforward two-part process. The image files need to be restored from backup, and the catalog needs to be restored from backup. If necessary, previews can be regenerated automatically, and any information from the catalog can be pushed back into the newly restored files.

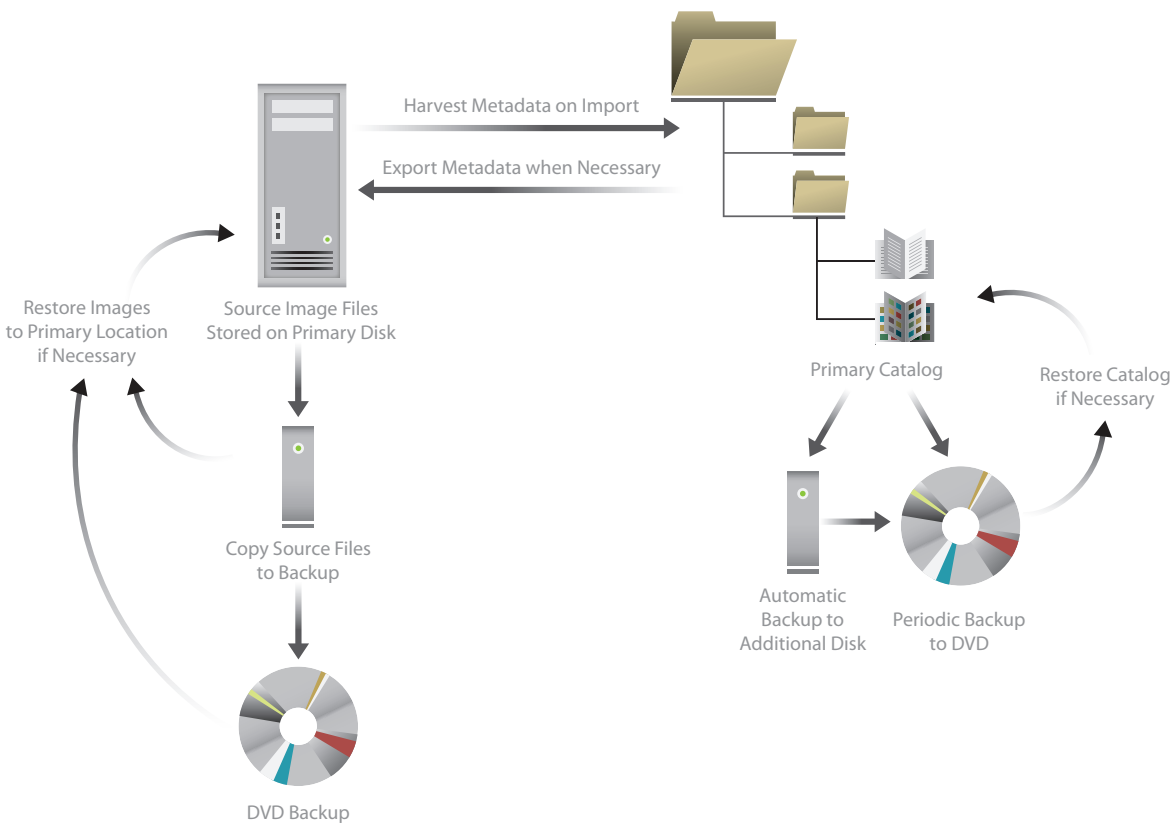


Figure 16: One of the main benefits of the one original structure of cataloging software is the simplification of backup and restoration procedures.

Generates multiple output types from a single source

One of the great advantages of catalog-based NDI software is the ability to generate many kinds of output from the same source image. Because all the information about the photos can be gathered in a single environment and can be integrated with the ability to control them, the user has unparalleled capacity to group and make use of photos on a collection-wide basis.

There can be many different ways to organize a collection of photographs. It might be useful to group photos according to subject matter, client, usage, or quality rating. And once these groupings are created, one might wish to output a subcollection to print, a slide show, a web gallery, or another derivative product.

Catalog-based PIE software that is integrated with a rendering engine can create these derivative products even more efficiently than non-rendering cataloging software. It's frequently desirable to do a bit of touch-up to the image files on the way to output. Having that ability within the same program that groups and sends files to output lets the user integrate picture selection and image editing in a streamlined way.

Conclusion

Non-destructive imaging has seen a great increase in capabilities and user acceptance over the last couple of decades, and that trend is likely to continue. While non-destructive imaging offers some hurdles—the process requires a bit of a learning curve—the advantages are clear in terms of cost savings, time savings, and creative freedom.

In the world of digital photography, catalog-based NDI systems are capturing lots of attention, and rightly so. While the functions they enable will continue to increase in amazing ways, the basic structure—a central catalog referencing a library of separate source images—is likely to stay relatively constant. Understanding the components of catalog-based non-destructive imaging systems enables you to make best use of these powerful tools.

Glossary

Catalog-based PIE software: Catalog-based PIE software is cataloging software that also includes a parametric rendering engine.

Cataloging software: Cataloging software is a kind of database software that keeps track of images and all the informational metadata about the images.

Default rendering: The default rendering is the way the image is displayed when it is first loaded into PIE software.

Derivative file: A derivative file is a copy of a source image that is saved out to a separate file. A derivative file may or may not have had explicit rendering or processing changes baked into it, relative to its source image.

Engine-dependent rendering: An engine-dependent rendering is a version of an image that is dependent on the rendering engine to be created.

Fixed rendering: A fixed rendering is a rendering of the image that is saved into pixel values or printed on paper. In a color-managed workflow, it is not dependent on any particular rendering engine to display correctly.

Informational metadata: Informational metadata is information that describes the content, ownership, or usage of the image file. The most common examples of informational metadata generally fall into EXIF or IPTC categories.

Live rendering: A live rendering is a view of the images that only exists when the image is loaded into PIE software.

Non-destructive imaging (NDI): Any system of image adjustment that can be accomplished without changing the original source image.

Parametric image editing or processing instruction editing (PIE): PIE is image adjustment by means of rendering parameters. Instead of changing the original source image data, images are sent through a rendering engine according to the user's settings. These settings, or parameters, can be saved as metadata.

PIE software: PIE software (also known as PIEware) is non-destructive software that adjusts images through the use of rendering parameters and a rendering engine.

Preview: A preview is a semi-permanent fixed rendering used to speed up the display of an image or to make the PIE rendering available to applications that don't own the same rendering engine.

Rendered image: A rendered image is the result of image adjustment after the image has been sent through a rendering engine.

Rendering engine: A rendering engine is the core software that performs image adjustments.

Rendering metadata: Rendering metadata refers to the instructions that control the rendering engine. Rendering metadata can include both the default rendering parameters for an engine as well as user-adjustable values for parameters, such as brightness or contrast.

Rendering parameters: Rendering parameters are instructions for adjustment and display of image files.

Saved engine-dependent rendering: A saved engine-dependent rendering is a combination of a source image, saved rendering metadata, and a rendering engine. It can be recreated by loading the image back into the rendering engine along with the saved settings.

Source image: A source image is the original image. Most often this will be a camera raw file, but a source image can be any file that is treated non-destructively as a source for subsequent usage.



ABOUT THE AUTHOR

Peter Krogh is a photographer and author specializing in digital photography. For 25 years, he has provided photographs for magazines, corporations, and agencies worldwide. His 2005 book, *The DAM Book* (O'Reilly) is the most widely read book on digital photography organization. He is on the board of ASMP. You can see his photography work on his website, www.peterkrogh.com.

Adobe Systems Incorporated
345 Park Avenue, San Jose, CA 95110-2704 USA
www.adobe.com

Adobe, the Adobe logo, Lightroom, and Photoshop are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners.

© 2007 Adobe Systems Incorporated. All rights reserved.
12/07