The Basics of Matchmoving

Anytime computer-generated imagery (CGI) needs to be placed into a live-action sequence, or vice versa, a matchmove is required. But what exactly is matchmoving? Matchmoving is the process of matching CG elements into live-action footage. As a result, it’s a crucial part of many visual effects shots. Despite its importance, it is completely invisible in the final shot—that is, if it’s done right.

In this chapter, I explain the key steps of a matchmove and how matchmovers work with the rest of the visual effects team. I’ve also included a tutorial that will help you become comfortable working with cameras and perspective.

Chapter Contents
Exploring a Typical Matchmove
Moving from 3D to 2D and Back Again
Understanding the Matchmoving Process
Matchmoving in the Production Pipeline
Perspective Matching Tutorial
Exploring a Typical Matchmove

To better understand what matchmovers do, let’s consider a typical visual effects shot. Let’s say the director has called for a CG creature to crash out of a building’s window and run across the street and into an alley. Because the monster will need to interact with the window, the visual effects supervisor decides that in addition to the monster, the window-shattering effect should also be done on the computer.

On the day of shoot, the director makes artistic decisions as to how to shoot the scene and eventually decides on a camera position. There is an opening in the building where the window should be, although the panes of glass are missing. The director and the cameraman practice the camera move a few times and watch the video playback to see how it looks. When they are filming, they move the camera as though it were following the monster crashing through the window and running across the street, even though the monster isn’t there. Extras react to the imaginary beast, and props around the window are rigged with monofilament string (fishing line) to be pulled down on cue as though they were knocked over. When the director is happy with the shot, the film is sent off to be digitized and then given to the visual effects artists to add the monster.

When the visual effects studio receives the digitized sequence (known as a plate), an animator will need to animate the creature and a technical director (TD) will need to create the glass-shattering effect. And, of course, a matchmover will have to match-move the plate.

The visual effects artists’ goals are to make their 3D elements look as realistic as the scene that was filmed. The animator needs to make the creature move as though it were really crashing through a window, and the TD needs to make the window shatter like a real window. The matchmover needs to figure out where the camera was and how it was moving when the scene was filmed.

Matchmovers play an important role in this case, because in order for the creature and window to appear matched realistically with the scene, they need to make sure that the CG objects are “filmed” the same way with their CG camera as the real set was filmed with the real camera. Consider the window that needs to shatter—if the perspective of the window doesn’t match the perspective in the plate, the CG window will look out of place. Furthermore, if the real camera moves to the left but the CG window stays put, everyone will know it’s a fake.

In our example effects shot, the on-set matchmover or data wrangler measures key items on the set. For example, she measures the size of the opening of the window as well as its height off the ground. She measures the distance across the street and the size of the opening to the alleyway. She draws a rough picture of the set and makes notes about positions of certain props and lights that might be useful to know. She also measures how high the camera is off the ground, what lens is used, and how far the camera is from the window.
Typically, the animator, TD, and matchmover all start working at the same time. Because some measurements have been made of the set, the animator knows how high the creature needs to jump to get through the window and how far to make it run across the street. The TD knows the size of the window that needs to shatter and how high it is off the ground. This is enough information to allow them to start setting up their scenes.

While they’re doing that, the matchmover starts by first examining the footage to get an idea of how the camera was moving during the shot. He brings the footage into his matchmoving software and begins to track the 2D features in the scene as they move around the screen. 2D tracking usually involves identifying things in the scene that don’t move (such as the corner of a building) and then letting the software follow that feature as the footage plays.

After the matchmover has tracked a number of 2D tracks, the software analyzes these tracks and computes the position of the camera in relation to the items in the scene. At the end of this process, the matchmover exports a scene to his 3D-animation package that includes an animated camera and the 3D positions for all of the features he tracked.

When the matchmover is satisfied with the camera he has generated, he goes about fitting that camera into a CG scene that is the same size as the one the animator and TD are using. When he’s finished, he is able to look through his CG camera at the CG window and creature, and they appear in the right perspective and scale when compared to the original live-action footage.

Finally, he saves the scene with the matchmoved camera in it. The animator and TD both use this camera to render their animations. If they’ve placed their window and creature in the right place in the environment, they don’t need to worry about whether the perspective matches or whether movement of the camera is the same. As long as they’ve rendered their CG elements with the matchmoved camera, it will appear matched into the footage.

Figure 1.1 shows an even simpler scenario where a CG car is inserted into a plate. The matchmover’s job is to work out where the camera needs to be. When the CG car is rendered through the matchmoved camera, it appears as though it belongs in the scene.

Figure 1.1 In the original “clean” plate (left), we see the image before the CG elements are placed in the shot. In the 3D-animation software, a CG camera is positioned in the environment (middle) in such a way that when the CG vehicle is viewed through it (right), the perspective matches that of the plate.
Moving from 3D to 2D and Back Again

No discussion of matchmoving would be complete without discussing cameras. Matchmovers need to understand how real-world cameras and lenses work and how they make the images we see on the screen. I discuss cameras more in-depth in Chapter 6, “Cameras,” but for now you need to know a few basic concepts about real-world cameras before getting started.

When a real camera films a scene, it is basically doing one thing: capturing the three-dimensional world as a two-dimensional image. That is, the camera gathers light from the 3D world around us and records it in a 2D form, such as a piece of film or a digital image. Let’s consider for a moment exactly how this happens.

The light from the scene passes through the camera lens and is focused onto film that rests in the film gate on the back of the camera’s inner chamber. The shutter closes, the film advances, the shutter opens again, and the process repeats. In the case of digital cameras, the film and film gate are replaced by a charge-coupled device (CCD) or complementary metal-oxide semiconductor (CMOS) chip that electronically captures the light information and records it to some sort of memory device.

The cameras in a 3D-animation program are based on real cameras but are represented in a slightly different manner. 3D-animation cameras represent a mathematically perfect model of the optics and construction of a real camera and lens. Like real-world cameras, they have a focal length and a film back (the equivalent of a film gate). But rather than capturing light from the real world, they are simply capturing information about the synthetic, computer-generated environment in which they have been placed.

Whether you’re dealing with exposed film or a 3D render, the resulting image is a projection. That is, the three-dimensional scene is flattened out into a two-dimensional representation of that scene. We have become so accustomed to these flattened images that we hardly notice them anymore, but every time we watch TV or a film, we are watching a flat recording of a three-dimensional scene (Figure 1.2).

Figure 1.2  When a camera captures an image of the three-dimensional world, it is really capturing the light from the scene that is projected across its image plane.
Understanding the Matchmoving Process

So if a camera’s purpose is to take the three-dimensional world and make a two-dimensional image, a matchmover’s job is the exact opposite. A matchmover must take a two-dimensional image and create a three-dimensional world. The portal between these two halves is the camera. If information about the camera can be reconstructed, it will go a long way toward helping the matchmover figure out how the 3D environment must have been set up at the time of filming. This information—details about the 3D environment and the camera—is what a matchmover will ultimately deliver to the animators to work with.

A matchmover’s workflow generally follows the same pattern for each shot, although there are a variety of ways to complete each task. Figure 1.3 shows a typical matchmoving pipeline. In the following sections, we’ll examine each of these steps more closely.

Figure 1.3 Matchmovers receive data from various sources, use that information to solve and set up the scene, and then deliver a scene with a matchmoved camera.

Evaluating the Footage

Evaluating the footage is perhaps the most important step in the matchmoving process, and unfortunately it is often overlooked. There are many ways to solve a matchmove, and careful scrutiny of the plate can help decide the appropriate tool to use, the pitfalls
to watch out for, and the length of time the process might take. This last item is particularly important, because clients and supervisors often put the timeline at the top of their list of questions.

One reason the evaluation process is so often glossed over is that many of the factors that determine the difficulty of a matchmove require some experience to judge. Some typical questions asked during the evaluation of a shot could include the following:

- What does the camera seem to be doing? Is it moving, and if so, how? Is it locked off or panning? How fast is it moving?
- What is visible in the shot? Are there tracking markers? Is anything blocking the markers?
- What format is the plate? Was it shot on film? DV? HD? Is there excessive compression, grain, or noise on the images?
- What needs to be placed in the shot? How accurate does it have to be?
- Who will be using the matchmove, and how will they be using it?

Of course, these only scratch the surface, but the more questions you ask, the more you will know what you need to do. The tutorials in this book will help you learn what these key questions are and how to deal with their implications. A more thorough list of questions is included in Appendix A, “Resources.” The Evaluation Checklist there can be used as a guideline to help determine the difficulty of a matchmove.

**Applying Information**

As I’ve said, solving a matchmove can be like solving a puzzle (it’s no coincidence that it’s referred to as solving a matchmove). And as such, the more information you have, the easier it should be to achieve good results.

The amount of information given to matchmovers can run the gamut. There might be only an image sequence and nothing else, or perhaps someone was allowed on set to record all the camera information and take measurements. Usually, it’s somewhere in between. But the good news is that a surprisingly small amount of data can go a long way toward solving the matchmove.

The following are typical data a matchmover might include:

**Camera Information**  This might include focal length, aperture, and film type.

**Set Measurements**  These can include camera height, focus distance, and measurements of various items in the shot.

**Survey Data**  This is a very detailed measuring of the set, usually done by a professional surveyor.

**Defining the Camera**

As stated before, the matchmover’s job is to define all of the internal and external parameters of the camera, and there are many ways to do that. Knowing which method
to use and under what circumstances comes with experience, but sometimes the only way to solve the matchmove is to experiment and see what works best. In broad terms, there are two major ways of solving for the camera: manual and automatic.

The manual methods harken back to the days before software existed to help matchmovers. This category includes perspective matching (matching the perspective of a single background image, rather than an image sequence, which is covered later in this chapter) and old-fashioned hand-tracking. In this method of tracking a sequence, the matchmover speculates as to the camera’s position and then refines it over many iterations until a match is achieved. Tracking a camera by hand is no small feat. It can often take weeks to truly figure out what is happening, because the process is nothing more than making educated guesses and tweaking them until they work.

Now there is software that allows a matchmover to track cameras somewhat automatically by using a sophisticated technology known as photogrammetry. These software packages (which are covered in the next chapter) all have a similar workflow. First, features in the image, such as props in the scene or tape markers (commonly used on blue-screen footage to mark points on the wall), are “tracked” as they move around the image in 2D. Then the software performs a calibration (or solve) for the camera by mathematically analyzing the movements of the 2D tracking markers. These packages generate an animated camera and 3D markers or nulls that represent the three-dimensional location of features that have been tracked in 2D. Matchmovers use this method most often because it is the easiest way to achieve a solution.

Some methods borrow from both manual and automatic techniques. Often these are customized solutions that revolve around both types of workflows. For example, the 2D tracking information from matchmoving software could be used with a custom script that allows the matchmover to solve for pan shots in a 3D application.

Fitting the Set

Although cameras are of primary concern, they are only half of the matchmoving process. Matchmovers must not only uncover all the facts about the camera, but also reconstruct the spatial layout of the environment seen in the live-action plate.

Figure 1.4 shows why this information is important. In order to properly match the CG building to the building seen in the image, we must find the correct placement for both the CG camera and the CG building. The first image shows an incorrect CG camera and building placement—what a mess. The second image has the correct camera placement, but the building is too close to the camera; therefore, the CG building doesn’t match the building in the image. But notice the third image. In this case, the building is the correct position and distance from the camera, but because the camera isn’t placed correctly, the CG building still doesn’t line up to the image. The last image shows how the scene looks with the correct camera and building placement. Because we have reproduced a CG camera and building that match the real-world camera and building, the CG building appears to match the building in the image. These images
illustrate how matchmoving is not just solving cameras, and not just solving environments, but also figuring out the relationship between the two.

How much of the environment does the matchmover need to reproduce? That depends on what is being placed into the footage. If the CG element is simply a character walking by, the animators and TDs might need only a simple ground plane. Other scenes might need rough geometry in order to cast 3D shadows. In some cases, such as digital set extensions, the matchmove might require an extremely accurate camera, detailed geometry, and spot-on positioning. Before beginning a matchmove, it is important to find out what type of 3D object is going into the scene and exactly where it will need to be placed.

3D environments might come from a variety of sources. Often matchmovers create the rough geometry themselves or are provided a set to “fit” into the plate. And in some situations, the matchmover may provide a rough set from which a more detailed set is later constructed by a modeler. Many times, matchmovers use 3D markers they’ve calculated during the matchmove to provide information about the spatial relationships of the scene. But regardless of where the information comes from, it is generally the matchmover’s responsibility to establish the environment and set up the scene so that other artists further down the production pipeline don’t need to worry about it.
Testing the Matchmove

After the matchmove is solved, it needs to be tested for accuracy. A bad matchmove usually shows up as an obvious disconnect between the live-action plate and the CG elements. For example, the CG element seems to follow the motion and rotation of the live-action scene, but then suddenly the CG element pops to another location or gradually drifts away from the feature it’s supposed to be resting on. Testing the matchmove consists of compositing the 3D objects over the image sequence and watching as the sequence moves to see whether there are any unusual pops, drifts, or jitter.

Most often, matchmovers will place low-resolution objects, or proxy objects, in the scene to help them determine the quality of a matchmove. One of the best ways to see whether an object is slipping is to place a checkerboard texture on it and render it out. This helps highlight slippage in 3D space.

A thorough testing is crucial, because a bad matchmove conceivably could work its way all the way through the pipeline and not be noticed until the end. And by that time, you would have a lot less time to deal with the problem and a lot more pressure.

Delivering the Scene

Last but not least, the final scene is delivered to other artists down the line. This step will vary greatly depending on whether the scene goes to a single artist working alone, or the scene is being turned over in a large production pipeline.

Certain items need to be considered, such as the following:

- Orientation and scale of the scene
- Objects that need to be included such as sets and characters
- Naming conventions, formats, and so forth

A well-organized and clearly laid-out scene will make other artists’ jobs easier and also make it less likely that you will have to explain the scene to someone after the fact.

Note: I’ve included a sample Scene Delivery Checklist in Appendix A. This checklist represents items I’ve found helpful in making my final matchmove scenes as foolproof as possible. I also cover this in more detail in Chapter 7, “Set Fitting.”

Matchmoving in the Production Pipeline

So how does matchmoving fit into the production pipeline? It really depends on the size of the production and the people who are being asked to provide the matchmoves. On larger-scale productions, a matchmove department deals specifically with matchmoving the shots. In smaller companies, one person may be handling the entire shot. Of course, matchmoving is necessary only if you are trying to fit CG elements into live-action plates.
Matchmoving usually happens early in a production. This is because the animators and TDs need to know exactly where to place their characters, explosions, and so forth, and they need a camera through which to look at and render their objects. For tight deadlines, matchmovers might also be asked to set up temporary cameras that approximate the move so that the TDs and animators can start their preliminary work and testing while the matchmove is being finished.

Figure 1.5 shows a fairly common production pipeline and where matchmoving fits in. The matchmover takes the original footage at full resolution, solves the matchmove, and then delivers a 3D scene to an animator or TD. In some studios, matchmoving data might be useful to other departments such as rotoscoping, paint, modeling, or compositing. (See Chapter 12, “Multipurposing Matchmoving Data,” for more on this.)

![Figure 1.5](image)

**Figure 1.5** In a typical pipeline, a matchmover delivers a scene to an animator or TD, but the matchmoving data can also be used by other departments.

**Perspective Matching Tutorial**

Now that you know how matchmoving fits into the big picture, you’re ready to get down to actually working on a shot. In this tutorial, you’ll create a simple
matchmoving scene that solves a problem many CG artists have faced at one time or another: perspective matching. In this scenario, we’re adding a CG UFO to an image of some buildings. In order to make the UFO feel like it belongs in the image, we need to create a CG camera that matches the real-world camera that shot the picture. We’ll also need to create CG buildings to represent the buildings seen in the image in order to check our perspective and to serve as objects with which the UFO can interact.

Perhaps the simplest form of matchmoving is matching the perspective of a single still image. Often these still images are used as backdrops for CG elements. Because the camera is not moving, all that needs to be done is to correctly position the CG camera relative to the CG object so that the perspective of the model looks appropriate. Even so, it can be frustrating to try to line up the CG geometry.

A few simple pieces of information can help make this process easier and less time-consuming:

- The focal length of the camera used to take the photo
- The height of the camera when the photo was shot
- The distance from the camera to an object in the real scene
- The scale of the object in the scene

Ideally, the person who took the image would provide all of this information. For this tutorial, you’ll need the image called 01_buildings_bg.jpg (shown in Figure 1.6). You can access the file from the Downloads tab on the book’s web page at www.sybex.com/go/matchmoving. For the purposes of this tutorial, let’s assume a worst-case scenario: We weren’t given any information about the camera or environment for this image. The best way to deal with these situations is to approach the problem methodically. By doing some digging, making careful estimations, and maintaining a 3D scene that is similar to the “real world” scene, we can zero in on a solution.

**Note:** For the tutorials in this book that use a 3D application, I use Autodesk Maya, but the steps are presented in a generic way so that you can use the program of your choice.

**Gathering the Data**

Let’s take a look at the image in Figure 1.6, in which we are asked to provide a CG camera and scene for the animators to work with. Consider the information we have and what we can do with it:

**Focal Length** The focal length of the camera is a little bit tricky to work out by eye. For a still image, it’s probably not necessary that this number be perfect, simply in the
ballpark. Until the building is in the right place, there’s no way to play with this value, so it’s often necessary to make an initial guess and then try to zero in on a usable value later. Fortunately, there are other ways to determine the focal length, which we’ll discuss later.

![Figure 1.6](image)

**Figure 1.6** Our task is to create a camera so that 3D objects can be placed above the buildings.

**Measurements**  There aren’t any measurements of the scene, but perhaps the strongest assumption that can be made about this image is that it was taken by a person standing at street level. Therefore, it’s reasonable to assume that the camera is a little less than 2 meters off the ground.

**Note:** This book uses metric measurements, because that is the standard practice in most studios.

**Distance**  We don’t have any idea of the distance to the buildings or the scale of the buildings themselves. If we were given a model that was based on the blueprints of the building, we would have some solid data to work with, but because we don’t, we’ll have to make a guess at this as well.
**Exif Data**

Most digital cameras can record information about the settings of the camera and lens and embed it in the image itself as Exchangeable Image File Format (Exif) data. The type of information captured varies from camera to camera but can include the camera’s make, aperture, focal length, shutter speed, and even the Global Positioning System (GPS) coordinates where the image was taken.

Exif data can be embedded in JPG or TIFF images and read by most image-editing and image-viewing programs. It can also be viewed in Windows (right-click the file and choose Properties) and on Macs (Finder > Get File Info).

Because the data is embedded directly in the image, the data is preserved as the file is copied from one place to another. Be careful if you are modifying the image in an image-editing program such as Adobe Photoshop—unless you are careful, resaving the file could result in the loss of this information.

---

**Setting Up the Camera**

As is often the case with a matchmove, the process is not a linear one. Usually, it is an iterative one, as we do one or two things to get a rough result and then make a few more passes to refine the camera and objects and build on known facts until at some point we have made the match to the required level of accuracy. This example is no exception. The following are the steps to get the building matched:

1. **Evaluate the scene.** Because we’re not dealing with images in motion, the key thing to figure out when matching the perspective of an image is where your 3D models need to go. Let’s say that we want a spaceship to fly above the buildings and cast shadows on them as it goes by. That means we need to know the camera’s position as well as the position of each of the buildings.

2. **Make a camera and create an image plane or background image that uses the still image.** First we need a camera to work with. This should be a freely moving camera (as opposed to a targeted or aimed camera). Because the ultimate goal is to align the CG objects with the plate, we will need to set up the camera or environment so that the image is seen as a backdrop behind the 3D objects when we look through the camera.

3. **Plug in the focal length for the camera.** The two main values we’ll need to know for the camera setup are the focal length and the film back. We have no notes from when the picture was taken, but fortunately, we can examine the Exif data contained in the JPG file to determine the focal length. Most image-viewing and image-editing programs have the capability to display the Exif data for an image if it contains any. (See the sidebar “Exif Data.”) For example, by choosing File Info in Photoshop, we see the Exif data that lists the focal length used for this shot: 18 mm (Figure 1.7).
4. **Enter the film-back values.** The Exif data also tells us the make of the camera, in this case a Panasonic Lumix GF2. A quick Internet search reveals that the film-back size for this camera is 0.681” × .512” (17.3mm × 13mm). If you don’t know the film back, it’s best to go with a standard preset value or use the default settings in your animation program. The reasons for these values are a bit complicated, so I’ll save the explanation for Chapter 6, but at least this will give us something to work with. For now, it’s important to know that if you don’t know the film back, having the focal length listed in the Exif data might not be all that useful.

5. **Set the height of the camera.** Because this is the strongest assumption we can make about the camera, we’ll enter it first. We’ll assume an average height for the camera of about 1.75 meters off the ground, so place the camera at 1.75 m up on the y-axis.

---

**Not All Film Backs Are the Same**

Don’t worry if you find yourself a little bit confused by all of this film-back business. It can bewilder even the most seasoned visual effects pros. Often 3D artists just guess at the value or use preset values in the software.

To make matters even more confusing, every 3D animation and matching program uses different terminology and has different ways of entering the information. Matching programs use the term film back, whereas most 3D animation programs call this the aperture. For example, Autodesk 3ds Max calls it the Horizontal Aperture and allows you to enter only the horizontal measurement in millimeters. NewTek LightWave uses the vertical aperture only, and Maya allows you to enter both horizontal and vertical aperture measurements, but only in inches! You should consult your software’s documentation to see where you need to enter this information and how.
Adding Rough Geometry and Refining the Camera

Now that we have a camera set up, we need to start fleshing out the geometry in the scene. When the animators get ready to put the UFO above the buildings, they’ll need to know where the buildings are. The task of identifying the location of objects in the scene often falls on the shoulders of matchmovers. This geometry needs to be accurate to the image so that any objects that are added to the scene will appear to be in the correct location. You usually don’t need to build geometry for the entire scene, just the portions that will interact directly with the CG objects.

I’ve found that creating geometry for a scene can be done in conjunction with the initial camera setup. Often you’ll need to make continual refinements to the camera and geometry until you’ve zeroed in on the correct placement for both. The good news is, after you’ve built one or two objects, the rest become much simpler to build. Also, after you’ve built objects to work with, finding the camera’s exact position becomes much easier.

Before you get started on the next steps, you may find it useful to split up your viewport so that one view is looking through the camera, while other views are looking at the scene from a top, side, or perspective view. This way you can “see” what the camera sees while you are adjusting things in other views.

1. **Create a building of an estimated height.** This is where it starts to get tricky, because we really have little to go on here. I estimated the building size by looking at the windows. I figured that each floor of the building was about 4 meters tall. So I counted the number of floors and multiplied it by 4 m. I guessed that the building on the right was 18 floors, so 72 meters seemed about right. Likewise, I tried to use visual clues to estimate the width and depth of the building. After my initial guess, I came up with values for my building of $x = 40$ m, $y = 72$ m, and $z = 40$ m. Because the building in the photo has 18 floors, it would be helpful to also add 18 subdivisions to the geometry’s height.

2. **Move the building to an estimated distance away from the camera.** This is likely to be the most difficult value to estimate. There aren’t too many clues as to how far the photographer was away from the building, so I needed to guess how far away the building should be. I chose 149 meters by watching the size of my building as I moved it away from the camera. This, of course, assumes that the size of my CG building is correct. If not, we can always adjust this later (Figure 1.8).

3. **Adjust only the rotation of the camera until the corner of your object is lined up with the forwardmost corner of the building in the image.** Now that our camera is roughly positioned and set up, we can start the business of lining up the object. Rather than trying to randomly match the entire object at once, it’s easier to try to line up just one point on the model to “lock down” one feature. This creates a connection, or common point, between the 3D scene and the 2D scene and creates a foundation on which to build a better match (Figure 1.9).
For this image, the camera is at its correct height, and the CG building is approximately the same scale as the building on the right.

By rotating the camera, we can get the upper-right corner of our 3D building to match up with the image.

For now, don’t concern yourself if the perspective or scale of the object is off. Just focus on getting one corner of the 3D building to match with the corner of the building on the image. The idea here is to find one common point between the CG object and the real-world scene. If we can get one corner of each aligned properly, we can use that as the basis for aligning the rest of the scene. For now, we can assume that the camera height and distance are correct, so we don’t want to translate (move) the model. In my scene, I simply rotated the camera until the upper-right corners lined up.

Creating a Camera Rig

Throughout this book, we’ll be creating camera rigs in our 3D applications. A camera rig is similar in concept to a real-life camera rig such as a dolly or a crane. It simply is
another way to move the camera around more conveniently in the scene. How these rigs are built will vary between the different software packages, but they generally consist of grouping or parenting the camera under one or more nulls (or similar nonrenderable objects) so that the camera can be moved around a different pivot point.

In this case, we have aligned the corners of the CG building and the building in the image, thereby establishing a link between the two. We want to be able to move the camera while maintaining that alignment, so we'll create a rig that does exactly that. Here are the steps to create and position a camera rig:

1. **Make a parent object for the rig.** Create a null object in the scene and name it CameraRig. Snap the null to the upper-right corner of the 3D building that we lined up in the preceding section.

   **Note:** Null objects go by various names in different 3D packages. For example, in Maya they’re referred to as “Locators,” and in 3ds Max they’re called “Dummy Objects.” Basically, whenever I use the term “null,” I mean a 3D object that doesn’t render and can be used in a parenting structure or to mark a 3D location.

2. **Parent the camera to the CameraRig null** (Figure 1.10). Check the parenting settings to make sure that it stays in its position during the parenting (as opposed to snapping to the position of the null). For example, in Maya if you open the option panel for the Parent command, you want to make sure that the Preserve Position check box is selected.

![Figure 1.10](image-url) The camera rig is created by creating a null that is positioned on the upper-right corner of the image (the object that looks like a crosshair). Then the camera is parented to the null.
3. **Use the camera rig to rotate the camera into position.** We can now rotate the CameraRig null to reposition the camera and try to match the perspective. We rotate it on the y-axis only because our strongest data suggests that the photo was shot at street level, and therefore we want to maintain that relationship. If the rig were rotated on the other axes, the camera would be raised or lowered off the ground, and that would be one less piece of information we could take for granted.

If after rotating the camera rig on the y-axis the building still doesn’t line up, you may have to adjust the rotation of the camera a bit more. This will cause the CG building’s corner to come away from the building on the image, so you’ll have to reposition it. You might have to go back and forth a few times between rotating the rig on the y-axis and rotating the camera to get the perspective working better (Figure 1.11).

![Figure 1.11](image)

*Figure 1.11* Note how the rig allows you to adjust the position of the camera simply by rotating the null. When you rotate the null on the y-axis, the camera moves around the null on the x- and z-axes. This keeps the corner of your 3D building on the corner of the building in the image while you adjust the perspective.

Camera rigs give you more freedom and control over how you can position the camera. You’ll use them quite often in the matchmoving process, and they can really help make it less frustrating.

**Evaluating and Adjusting the Camera**

Up until this point, we’ve been working with approximations for our camera and building. Now we have everything we need to “tighten” up the matchmove and find the exact settings for the camera and the right size and shape for the building. In this phase, we are trying to bring the scene into perfect alignment:

1. **Is focal length right? Experiment if needed.** Because we know the focal length used in this image, and that focal length seems to be working, we don’t need to adjust it. However, if we didn’t know the focal length, now would be the time to try to fine-tune it.
The focal length can be a little tough to guess, but wider lenses (smaller focal lengths) tend to look more distorted, while longer lenses (higher focal lengths) look flatter. You may need to make some adjustments to the building’s position at this point as well. It’s good to do this in tandem with adjusting the focal length, because both produce a similar result, making the building look larger or smaller in the frame.

2. **Make adjustments to building scale if necessary.** Now it’s becoming obvious that the building’s scale isn’t too accurate, so we adjust it by scaling it to fit better. When you scale the building, you should scale it from one of the bottom corners, because you want the bottom of the building to stay on the ground. If you scale the building from the center, it will be more difficult to control, and you’ll need to continually move it so that the bottom of the building is back on the ground. In most software packages, you can do this by moving the object’s pivot point to the bottom-right corner (directly beneath the CameraRig null) of the object (Figure 1.12).

![Figure 1.12 Rescaling the building from its bottom finishes off the perspective match.](image)

3. **Continue adjusting the camera rotation, group rotation, and object placement until a fit is achieved.** From here, it’s just a matter of tweaking the camera and/or building if necessary. One of the benefits of having created the camera rig is that it gives you several more degrees of freedom without losing accuracy. The CameraRig null allows you to spin the camera to different locations around the building (on the y-axis only). If you need to adjust the distance of the camera, you can move the camera itself along the x- and z-axes and rotate it on all three axes.

4. **Place a second building.** After we’ve figured out the camera and the position of the first building, the second building’s placement is easy. From the looks of
it, they are both aligned at their fronts, so we can simply duplicate the existing building and slide it down the x-axis until the right edge of the 3D object lines up with the right edge of the next building in the image. From there, you can scale the geometry until it matches (Figure 1.13). You can repeat the same process in order to place the other three buildings that we see in the photo.

Figure 1.13 The other buildings can be placed simply by duplicating the original, sliding it over on the x-axis, and scaling to match the other buildings.

With both buildings in place, you’ve provided a scene with the correct camera and enough information to show animators or TDs (or even yourself) where the CG object needs to go. If we were adding a UFO between the two buildings, as in Figure 1.14, our rough geometry would show us where to put it. These objects can also serve as geometry for reflections, shadows, or other effects.

Figure 1.14 Now we’re ready to have our UFO do some fancy flying between the buildings.
Moving Toward Moving Pictures

As you can see, matchmoving doesn’t have to be a completely blind process of random guesses. By building on the information at hand, you can infer the data you don’t have. Although we are working on a still image here, the same techniques can be applied to a moving sequence as well.

This method works well for these relatively simple still-image situations, but for more-complicated shots, it is useful to enlist the help of matchmoving software. Starting with the next chapter, I cover how these programs work, and how they can be used effectively to help solve even the toughest shots.