An All-In-One Solution to Geometric and Photometric Calibration *

Julien Pilet†
CVLab
École Polytechnique
Fédérale de Lausanne

Andreas Geiger
Universität
Karlsruhe

Pascal Lagger
CVLab
École Polytechnique
Fédérale de Lausanne

Vincent Lepetit
CVLab
École Polytechnique
Fédérale de Lausanne

Pascal Fua
CVLab
École Polytechnique
Fédérale de Lausanne

(a) (b) (c) (d)

Figure 1: From calibration to augmentation. (a) Our setup. Note the complexity of the ambient lighting. (b) A raw image acquired by one of the cameras. The planar pattern on the box is automatically detected and used for calibration purposes. (c) A virtual teapot has been added. It is properly lighted and casts a shadow on the real box. (d) The virtual teapot is still correctly registered and occluded by the box even though the calibration pattern is not visible in this view.

ABSTRACT

We propose a fully automated approach to calibrating multiple cameras whose fields of view may not all overlap. Our technique only requires waving an arbitrary textured planar pattern in front of the cameras, which is the *only* manual intervention that is required. The pattern is then automatically detected in the frames where it is visible and used to simultaneously recover geometric and photometric camera calibration parameters.

In other words, even a novice user can use our system to extract all the information required to add virtual 3D objects into the scene and light them convincingly. This makes it ideal for Augmented Reality applications and we distribute the code under a GPL license.

1 INTRODUCTION

We propose a simple-to-use camera calibration system that can handle multiple cameras whose fields of view do not necessarily overlap. It estimates the geometry of the cameras, their photometric responses, and an environmental lighting map. The only manual intervention required involves waving an arbitrarily textured planar pattern in front of the cameras. In other words, in one single operation, our system yields all the information required by sophisticated Augmented Reality applications to draw virtual 3–D objects at the right locations and then light them convincingly, as shown in Fig. 1.

In the remainder of the paper we will refer to the recovery of geometric properties as *geometric calibration* and the estimation of photometric responses and lighting parameters as *photometric calibration*. Since geometric camera calibration is now very well understood from a theoretical point of view [9], there are of course a number of practical techniques, including some commercial systems that are powerful but expensive. Most rely on retro-reflective targets [20] or laser pointers [18] that can be shown to the camera or cameras. However, these approaches require adjusting the shutter speed or camera aperture, which prevents simultaneous geometric and photometric calibration.

Our approach extends earlier methods [17, 21], whose implementation is now available in OpenCV [15]. They rely on a printed grid that is shown to the camera. However, the grid detection is less reliable than using markers [5]. By contrast, our method is more flexible since it works with any planar textured surface. Furthermore, unlike ours, these methods neither handle extrinsic calibration of non-overlapping fields of view nor any photometric information. A method that calibrates multiple camera is presented in [19]. It also recovers the relative poses of multiple cameras but requires each camera to see all calibration target positions, which, in practice, is a very severe limitation.

In our system, the user only has to move the calibration pattern in front of the cameras. We use a powerful computer vision technique to detect it in the images [12] both robustly and in real-time. This provides homographies between the pattern and the images, which are then used to compute the intrinsic camera parameters as in [21, 17]. However, unlike these earlier approaches, we also use them to compute the poses of the cameras with respect to each other. Finally, we rely on the intensity variations within the pattern and across the images to achieve photometric calibration. Our all-in-one approach includes the following contributions that both reduce the user's workload and give our system its additional capabilities:

- **Fully automated detection of the calibration pattern**. Our approach to detecting the calibration pattern is both real-time and robust. Furthermore, it automatically selects the frames that yield the most informative homographies and rejects ambiguous ones. As a result, it becomes easy to collect large amounts of calibration data, which in turn yields excellent numerical stability and accuracy without any extra manual intervention.

- **Handling one or more cameras with non-overlapping fields of view**. Our system selects the position of the pattern that is seen by the largest number of cameras to define the common referential. As depicted by Fig. 2, it then expresses

*This work was supported in part by the Swiss National Science Foundation

†Contact author. E-mail:julien.pilet@epfl.ch

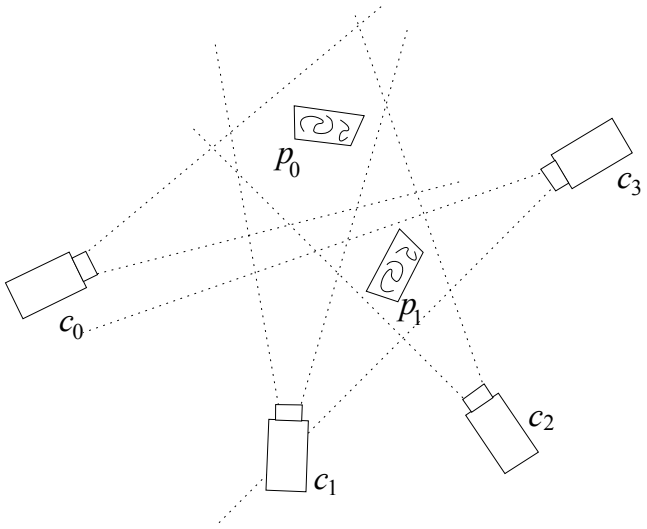


Figure 2: A camera network. In this example, the field of view of camera c_3 does not overlap with those of cameras c_0 and c_1. Nevertheless, a full registration is still possible. The displacement between cameras c_2 and c_3 can be estimated when the pattern is in pose p_1, while the relative positions and orientations of cameras c_0, c_1, and c_2 can be estimated using pose p_0. By chaining these estimates, we can express the external camera parameters for all cameras in a common referential.

the pose of the cameras that do not see this position by composing relative displacements between camera pairs. These are easy to estimate from the homographies and give initial estimates that are then refined by bundle adjustment.

- **Photometric calibration for free**. We use the very same set of images both for geometric *and* photometric calibration, which means that no additional user intervention is required to obtain the latter. We take advantage of the fact that our pattern effectively samples the space of surface normals to sample an irradiance map which we deconvolve to render virtual objects with correct specular reflections and cast shadows, such as those of Fig. 1(c) and 1(d).

Our system is embedded in a distributed framework that allows real-time processing of the output of several cameras plugged into separate but networked computers. To demonstrate its effectiveness, we distribute the code under a GPL license [1].

In the remainder of the paper, we first compare our approach to existing calibration techniques. We then introduce the techniques we use to recover first the geometric camera pose parameters and then the photometric and illumination ones. Finally, we evaluate the accuracy of our system and present Augmented Reality results.

2 Related Work

We review briefly existing approaches to geometric and photometric calibration, which are usually performed separately. Our approach is inspired by several of them but performs both simultaneously, which makes it easier to use for Augmented Reality purposes.

2.1 Geometric Calibration

Calibration algorithms that do not require an object known *a priori* are sometimes called *auto-calibration* algorithms. They usually involve a moving camera that is calibrated by simultaneously recovering pose parameters and reconstructing the 3–D structure of the scene. However, they are not well adapted to computing the relative positions of multiple cameras. Furthermore, they lack robustness.

We therefore focus here on methods that rely on a calibration object or pattern because they are much more reliable. Commercially available systems rely on tri-dimensional calibration objects with retro-reflective markers, spheres, or disks on them, which are often complex to build and cumbersome. A notable exception are approaches such as [13] that only involve a moving wand with two markers on it. However, all methods that depend on retro-reflective markers require changing the shutter speed to reliably extract the markers from the images, which prevents simultaneous geometric and photometric calibration. Similarly, using light patterns emitted by a laser pointer as in [18] is a very practical approach to geometric calibration but requires modifying the cameras settings.

Our system falls into the category of recent approaches that rely on a planar target moved in front of the camera [17, 21]. This is attractive because such a target can be built by simply printing a pattern on a sheet of paper. These earlier techniques, however, were only designed to recover the internal parameters of a single camera. Here we are also interested in the positions and orientations of the cameras with respect to each other. The method closest to ours that we are aware of is presented in [19]. As ours, it provides both the intrinsic parameters and relative poses of a multi-camera system using a planar target. However the parameters are estimated via factorization of a matrix built from homographies between image pairs. This requires that *all* the positions of the planar target must be seen from *all* the cameras simultaneously. This is a major limitation that our method does not have.

The papers discussed above focus on the geometric aspects of the problem, and underplay the pattern detection and the frame selection issues. In practice, the target position is often provided by hand to perform the experiments, which is inconvenient. [15] uses a black and white checkerboard as a calibration object to facilitate the point extraction procedure. However the repetitive nature of the pattern can easily result in ambiguous poses, and its detection is known to be error-prone. Here we solve this problem using a state of the art feature point-based method [11], which can deal with arbitrary textures and is reliable enough not to require manual filtering of misdetections.

2.2 Photometric Calibration

Real-world illumination can be captured using calibration objects such as reflective spheres. This can be done as a preprocessing step [3] if the illumination remains unchanged. It can also be done in real-time [10] using a calibration object built by adding a mirror ball to a 2–D square marker. While effective, this approach is much more difficult to deploy than ours since constructing such an object is much more involved than simply printing a textured pattern. Similarly, [16] relies on omni-directional stereo cameras, which requires specialized hardware instead of the ordinary cameras we use. The photometric calibration we get may not be as accurate as those obtained with such lighting probes but we will show that it is amply sufficient to synthesize realistic augmented images while being much more light-weight.

Another class of approaches [4, 6] focuses on interactive rendering of illumination changes caused by virtual objects in the real scene. However, this typically involves a 3–D scene model, which is cumbersome to acquire.

Our approach to creating an environmental lighting map is related to the inverse lighting framework of [14] that involves an object of constant albedo and known shape. If the object surface contains a sufficiently large number of differently oriented normals, it is possible to relate a lighting contribution to each direction of a discretized lighting sphere. Regularized deconvolution then allows the estimation of the light sources position. In our case, as the grid

moves in front of the cameras, it samples the spaces of normals. Since we precisely compute those normals, we can directly exploit the observed changes in pixel intensities to model the illumination using a very similar approach. This removes the need for a constant albedo 3–D object and its model.

3 GEOMETRIC CALIBRATION

Our geometric calibration procedure includes several stages. Our system first computes homographies between the geometric plane of the target object and its image projections. It then retains the most reliable ones and the corresponding frames to estimate the intrinsic camera parameters and the relative pose of the calibration object with respect to the cameras in the corresponding frames. In turn, these poses are used to select a common referential and to compute the positions and orientations of the cameras with respect to each other. Finally, our system performs global non-linear minimization to refine these estimates. We outline the individual steps of this process below.

3.1 Computing and Selecting the best Homographies

To compute the homographies between the geometric plane of the target object and its image projections, we must first detect it in as many frames as possible. To this end, we use an interest point-based method that we developed in earlier work [11]. It first performs real-time matching of interest points extracted from a reference image such as the ones of Fig. 3 against those extracted from input images acquired at run-time under potentially large perspective and scale variations. It then uses standard robust estimation techniques to compute homographies from these matches and selects the best.

3.1.1 Wide-Baseline Matching

We achieve fast and robust wide-baseline matching by formulating it as a classification problem, which lets us shift much of the computational burden to a training phase. We treat the set of all possible appearances of each individual interest point as a class, which we refer to as its *view-set*. During training, given a fronto-parallel view of the target object, interest points are extracted and numerous synthetic views of their possible appearance under perspective distortion are generated and used to train a set of randomized trees [2]. These trees are then used at run-time to recognize the distorted interest points by deciding to which view-set, if any, their appearance belongs. In theory, we could have used other kinds of classifiers for our purpose but we chose randomized trees because they are robust and fast, while naturally handling multi-class problems and being reasonably easy to train.

In this context, a simple and fast interest point detector such as the Harris corner detector [8] suffices for operation even under large perspective and scale variations, which results in a further performance increase. While earlier methods require detectors that can be depended upon to produce very repeatable results, which can be very time-consuming, the most repeatable object keypoints for the specific target object are simply found during the training phase.

3.1.2 Estimating the Homographies

Given enough correspondences, the homography between the target image and the input one is easily estimated from the correspondences by RANSAC followed by non-linear least-squares estimation. For each successful such computation, this gives us a homography $\mathbf{H}_{c\leftarrow p}$ such that:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{H}_{c\leftarrow p} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (1)$$

that maps each point [1] $[X,Y]^\top$ of the calibration pattern plane to its corresponding 2D point $[u,v]^\top$ in camera c for a given pose p. The scalar λ accounts for the fact that homographic transformation is expressed in homogeneous coordinates.

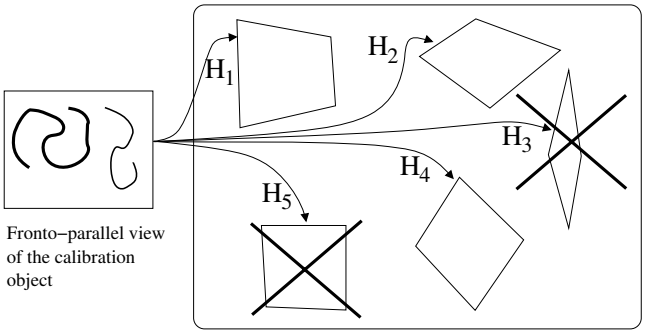


Figure 4: Dropping unreliable homographies. Homography $\mathbf{H}_3$ is ignored because, with such a slanted surface, interest points detection becomes inaccurate and error-prone. $\mathbf{H}_5$ is also discarded because it yields a calibration singularity. The other homographies can be safely kept.

3.1.3 Selecting the Best Homographies

Unlike checkerboard-based methods, our approach to estimating the homographies is sufficiently robust not to generate erroneous matches that would have to be removed by hand. What happens in practice is that when the pattern is either occluded or too slanted, it is simply not detected.

However, this is still not quite enough because some of these homographies are inherently ambiguous or singular from a calibration point of view. As shown in Fig. 4, these unreliable homographies come in two flavors. First, the ones that insufficiently distort the pattern are ambiguous and, depending on the noise, may yield wildly different pose estimates. Second, those that distort the pattern too much also produce unreliable pose estimates because the interest points become difficult to locate precisely enough.

To overcome this problem, we define a square in the plane attached to the pattern and measure the angle at each corner after warping it by the homography. If one of the angles is too large or too close to $\frac{\pi}{2}$, the homography is rejected. More formally, a homography $\mathbf{H}$ is rejected if at least one of the angles α of the warped square verifies

$$\begin{array}{l} \cos(\alpha) > \cos(\alpha_0) \text{ or} \\ \cos(\alpha) < \cos(\pi - \alpha_0) \text{ or} \\ \cos(\frac{\pi}{2} + \alpha_1) < \cos(\alpha) < \cos(\frac{\pi}{2} - \alpha_1). \end{array}$$

Good results have been achieved using $\alpha_0 = 0.01$ and $\alpha_1 = 0.005$. The filtered homographies are then sorted by the likelihood assigned by the robust estimator of Section 3.1.2 and only the best ones are kept. In practice, we retain around fifty for each camera. As will be shown in Section 5.1, this is enough to guarantee accuracy without imposing an unnecessary computational burden.

3.2 Initial Estimation of the Internal Parameters

The internal parameters are first estimated for each camera individually from the homographies $\mathbf{H}_{c\leftarrow p}$ using a method similar to the ones of [17, 21]. The computation is quite standard and is described in appendix for the sake of completeness. This yields for

[1] The symbol $^\top$ denotes the transpose operator throughout the paper.

(a) (b) (c)

Figure 3: Some examples of calibration patterns we used to test our system. As discussed in Section 3.1.1, these reference images serve to train a classifier. It is then used to match them against input frames and provide homographies to the geometric calibration process. The images are acquired under diffuse illumination so that the normalized pixel intensities are proportional to the albedo estimates required by the illumination model of Section 4.1.

each camera c a matrix of internal parameters

$$\mathbf{K}_c = \begin{bmatrix} \tau_c f_c & 0 & u_{0c} \\ 0 & f_c & v_{0c} \\ 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

where f_c stands for the focal length, τ_c for the aspect ratio, and $(u_{0c}, v_{0c})^\top$ for the principal point. These internal parameters of all the cameras will then be refined together with the external ones during the non-linear global minimization of Section 3.4.

3.3 Initial Estimation of the Poses

We recover the external parameters of each camera in a common referential in two steps. First, our algorithm computes external parameters in a coordinate system attached to the calibration pattern for each frame independently by making use of the internal parameters as estimated previously and the homography related to the frame. It then selects a common referential and computes camera poses in this referential by composing rotations and translations between pairs of frames.

3.3.1 Displacements between the Calibration Object and the Individual Cameras

Given a homography and the intrinsic parameters, we estimate the displacement of the calibration object with respect to the camera as described in the appendix. This step gives us the relative rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the calibration object with respect to the camera. Together, they represent the rigid *displacement* corresponding to pose p as seen from camera c. We write this displacement as the 4×4 matrix

$$[\mathbf{R},\mathbf{t}]_{c \leftarrow p} = \begin{bmatrix} \mathbf{R}_{c \leftarrow p} & \mathbf{t}_{c \leftarrow p} \\ 0 & 1 \end{bmatrix}. \quad (3)$$

The reverse displacement is computed by inverting the matrix which we denote as $[\mathbf{R},\mathbf{t}]_{p \leftarrow c}$.

3.3.2 Handling Non-Overlapping Cameras

In practice, the calibration object may never be seen by all cameras simultaneously. Our system therefore selects as a common referential the one attached to the pose of the calibration object seen by the largest number of cameras. It then expresses all the external camera parameters in this referential by composing the displacements of Eq. 3 and their inverses.

Fig. 2 illustrates this behavior. In this case, p_0 provides the common referential because it is seen by three cameras whereas p_1 is seen by only two. Even though camera c_3 does not see p_0, its external parameters in this referential can be estimated by composing

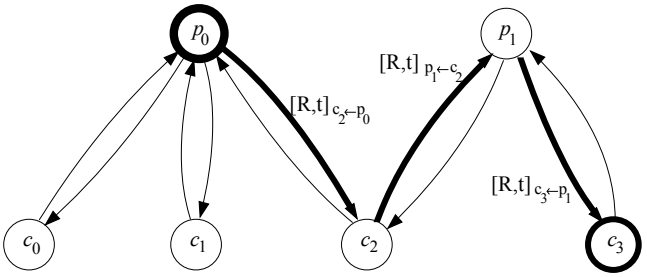


Figure 5: The graph corresponding to the network of Fig. 2. The common referential is defined by pose p_0, which is seen by three cameras while p_1 is seen by only two. The pose $[\mathbf{R},\mathbf{t}]_{c_3}$ of camera c_3 in this referential can be recovered by composing poses with respect to individual cameras. In this case, $[\mathbf{R},\mathbf{t}]_{c_3} = [\mathbf{R},\mathbf{t}]_{c_3 \leftarrow p_1} [\mathbf{R},\mathbf{t}]_{p_1 \leftarrow c_2} [\mathbf{R},\mathbf{t}]_{c_2 \leftarrow p_0}$.

the pose of p_0 with respect to camera c_2 with the displacement between cameras c_2 and c_3, which can itself be estimated from the poses of p_1 with respect to these two cameras.

Recovering such chains amounts to compute paths between nodes of a connected graph, which is well understood from an algorithmic viewpoint [7]. More specifically, we define a graph whose nodes correspond to the cameras and the poses of the calibration object, such as the one depicted by Fig. 5. An edge links a camera node and a pose node when the camera sees the pose and is labeled with the corresponding displacement.

Let p_0 denote the pose that defines the common referential, and $[\mathbf{R},\mathbf{t}]_c$ the pose parameters of camera c in this common referential. Since we have by definition $[\mathbf{R},\mathbf{t}]_c = [\mathbf{R},\mathbf{t}]_{c \leftarrow p_0}$, we first look for a path between the p_0 and c nodes, which we write as

$$p_0 \rightarrow c_{\sigma(1)} \rightarrow p_{\sigma(2)} \rightarrow c_{\sigma(3)} \rightarrow \dots p_{\sigma(n)} \rightarrow c\,.$$

where $\sigma(.)$ is a mapping function on the indices that defines the path. Note that the path alternates object pose nodes and camera nodes. It gives us a way to compute $[\mathbf{R},\mathbf{t}]_c$ from the displacements we just computed since

$$\begin{aligned}[\mathbf{R},\mathbf{t}]_c \quad &= \quad [\mathbf{R},\mathbf{t}]_{c \leftarrow p_0} = \\ &[\mathbf{R},\mathbf{t}]_{c \leftarrow p_{\sigma(n)}} \dots [\mathbf{R},\mathbf{t}]_{c_{\sigma(3)} \leftarrow p_{\sigma(2)}} [\mathbf{R},\mathbf{t}]_{p_{\sigma(2)} \leftarrow c_{\sigma(1)}} [\mathbf{R},\mathbf{t}]_{c_{\sigma(1)} \leftarrow p_0}\,.\end{aligned}$$

The $[\mathbf{R},\mathbf{t}]_p$ pose parameters of the calibration object can be estimated similarly.

Obviously, this will only work if the graph has one single connected component. In practice, assuming that no camera has a field of view that does not overlap at all any of the others, this is always the case if we move the calibration pattern sufficiently.

3.4 Refining the Estimation

Computing displacements by composing pairwise motions is effective but not particularly accurate. Therefore, to refine not only the

pose parameters but also the intrinsic ones, we minimize with respect to all cameras simultaneously the sum of the reprojection errors for the point correspondences used during the detection step of Section 3.1.2. This bundle adjustment is expressed as

$$\sum_{c=1}^{C}\sum_{p=1}^{P}\sum_{k=1}^{M(c,p)}\left\|\begin{pmatrix}u_{c,p,k}\\ v_{c,p,k}\end{pmatrix}-\text{proj}\left(\mathbf{K}_c,[\mathbf{R},\mathbf{t}]_c\,[\mathbf{R},\mathbf{t}]_p^{-1},\begin{pmatrix}X_{c,p,k}\\ Y_{c,p,k}\\ 0\end{pmatrix}\right)\right\|^2, \quad (4)$$

where

- C is the number of cameras, P the number of poses of the calibration object, and $M(c,p)$ the number of matches found by the detection stage for camera c and pose p. $M(c,p)=0$ if pose p of the calibration object is not seen by camera c;
- $[u_{c,p,k},v_{c,p,k}]^\top$ and $[X_{c,p,k},Y_{c,p,k},0]^\top$ are respectively are a 2–D point and a 3–D point matched by the detection step;
- $\text{proj}(\mathbf{K},[\mathbf{R},\mathbf{t}],\mathbf{M})$ returns the projection of 3–D point $\mathbf{M}$ under pose $[\mathbf{R},\mathbf{t}]$ and internal parameters $\mathbf{K}$.

To increase the robustness of our algorithm, we introduce a simple robust estimator in Eq. 4 to eliminate potentially incorrect correspondences.

3.5 Online vs. Offline Processing

All the computations up to the initial estimation of pose parameters of Section 3.3 are performed in real-time on an ordinary PC. The refinement stage of Section 3.4 is much slower and requires a few minutes for long calibration sequences. In practice, this means that we can run the system in real-time mode until the graph of Section 3.3.2 has indeed one single connected component and we can obtain initial estimates, which will then be refined offline. This makes the system extremely easy to use.

In real-time mode, the processing time varies from one frame to the other. It is therefore important to drop frames in a synchronous manner: If a camera drops a frame, all other cameras should also drop it. Since in our implementation the cameras are connected to several computers, we developed a networked solution to this problem: A process is attached to each camera and is considered as a network client that connects to a server. Every time a new frame arrives, the client informs the server and puts the image in a queue. The server either ask the client to accept or to drop the frame, depending on the other cameras and on the current computational load. If a frame is accepted, every client is informed and puts the image in a waiting-for-computation queue. At this point, the computation thread of each client, if ready, can start to detect the calibration pattern in the image. Upon termination, it sends the result to the server. Once the server has collected homographies and matches from every client, it can either accumulate data for calibration or, if the calibration has already been done, augment the images.

4 Photometric Calibration

Convincingly adding virtual objects into a scene not only requires proper registration, which is what the technique of Section 3 provides, but also photometric modeling so that they can be relighted correctly and properly blend into the environment.

As discussed in Section 2, most existing techniques perform geometric and photometric calibration as independent steps that require different camera settings. By contrast, our approach relies on the very same set of images to perform both kinds of calibration. For each camera, as long as the lighting and camera settings do not change, the pixel intensities within the calibration pattern depend only on its normal. In other words, each image in which the pattern is detected provides a number of samples corresponding to one individual surface orientation. Because we can easily and automatically collect many such samples, we can simultaneously recover the gain and bias of each camera and create an environmental lighting model that can be used for relighting purposes.

In this section, we first introduce the illumination model we use for complex environments such as the one depicted by Fig. 1(a). We then present two complementary approaches to instantiating it. The first one involves solving a linear system of equations derived from the calibration sequence to express the illumination as a function of surface normals. Once the gains and biases have been estimated, this model can be incrementally updated to reflect lighting changes. The algorithm can therefore be embedded into a real-time application that handles non-constant lighting. However, it is not designed to synthesize either shadows or specular reflections. We have therefore developed a second approach based on deconvolution. It explicitly computes a light distribution that could have produced the observed pixel intensities. It is more computationally intensive and makes no provisions for time-varying lighting but allows added virtual objects to cast shadows and produce realistic specularities.

4.1 Illumination Model

In a natural environment with extended light sources such as a room lit by rectangular windows, we *cannot* assume that we only deal with a small number of point light sources. Instead, we consider a potentially infinite number of sources that we assume to be both directional and outside the capture volume. For our purposes, this provides a satisfactory approximation of extended light sources.

By printing the calibration pattern on matte paper, we ensure that it reflects light equally in all directions. As a result, the difference in intensity values between images taken at the same time by two different cameras are due to shutter speed or aperture, but not to camera pose. If we further assume that the same sources are visible from every point of the calibrated volume, the amount of light reflected by a point on the calibration pattern depends only on the surface normal $\mathbf{n}_t$ at time t. To be robust to small localization errors, we do not consider individual points, but small patches π that average the local property around points on the calibration pattern. The irradiance x at surface patch π and time t can therefore be written as

$$x_{\pi,t}=\sum_{l=0}^{L}\max\left(\mathbf{n}_t\mathbf{d}_l,0\right)\Omega_l\,, \quad (5)$$

where Ω_l represents the radiosity, or power, of source l and $\mathbf{d}_l$ its direction. We assume a linear response for the cameras and write the pixel intensity $I(c,\pi,t)$ of patch π in the image acquired by camera c at time t as

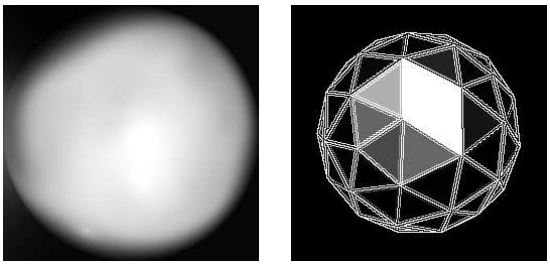
$$I(c,\pi,t)=g_c a_\pi x_t+b_c\,, \quad (6)$$

where a_π is the average surface albedo over π, g_c the camera gain, and b_c its bias. In practice these quantities can only be known up to a scale factor. We use for a_π the mean intensity over π in an image acquired under uniform diffuse lighting such as the ones of Fig. 3, which simply amounts to selecting a particular scale factor.

4.2 Online Lighting Calibration

Instead of explicitly evaluating the Ω radiances of Eq. 5, we directly compute the $x_{\pi,t}$ irradiances as a function of the orientation of the $\mathbf{n}$ normals. More specifically, we simultaneously compute the gains, biases, and $x_{\pi,t}$ irradiance values for the normals we have observed. We then interpolate this set to estimate the unobserved values.

This yields a *light map* $M(\mathbf{n})$ such as the one depicted by Fig. 6(a). When augmenting an image c with a virtual surface



(a) (b)

Figure 6: Illumination models. (a) An interpolated light map computed using the online technique of Section 4.2. (b) A dome of light sources whose individual powers are estimated using the technique of Section 4.3. In this picture, the intensity of a triangle represents the power of an individual light source.

whose normal is $\mathbf{n}$, the augmented pixel value is set to $I(v,c,t) = g_c a_v M_t(\mathbf{n}_v) + b_c$, where a_v is the virtual albedo. This lightmap rendering is easily done by the GPU and requires only a short OpenGL shading language program.

4.2.1 Linear Estimation of the Light Map

The geometric calibration process provides many surface normals $\mathbf{n}_t$ and pixel values $I(c,\pi,t)$. To solve for the unknown gains, biases, and radiances, we linearize the problem by replacing some variables in Eq. 6. Let

$$\begin{aligned} g'_c &= \frac{1}{g_c} , \\ b'_c &= \frac{b_c}{g_c} . \end{aligned}$$

For each patch in each detected frame, Eq. 6 can be rewritten as

$$-I(c,\pi,t)g'_c + a_\pi x_t + b'_c = [-I(c,\pi,t)\ 1\ a_\pi] \begin{bmatrix} g'_c \\ b'_c \\ x_t \end{bmatrix} = 0 . \quad (7)$$

Putting all these equations together yields a large but sparse linear system and we find our solution as the eigenvector associated to the smallest eigenvalue.

4.2.2 Incrementally Updating the Light Map

The above computation assumes that the lighting does not change during calibration. However, once the gains and biases are known, computing a new surface intensity x_u from a new observed pixel $c_{\pi,u}$ is trivial. Thus, if the illumination changes, new frames can update the light map at the same time it shades a virtual scene. Each frame can sample only one normal at a time. Thus, if the light changes suddenly, it is not possible to update the whole irradiance map at once. Instead, we locally update the irradiance around the measured normal and keep the old values for other normals.

Let $M_t(\mathbf{n})$ be the light map at time t To update its value for a surface of normal $\mathbf{n}$ with the recently computed sample x_{t+1} corresponding to the observed real surface orientation $\mathbf{n}_{t+1}$, we write

$$\forall \mathbf{n} : M_{t+1}(\mathbf{n}) = (1 - f(\mathbf{n}))\, M_t(\mathbf{n}) + f(\mathbf{n}) x_{t+1} \ ,$$

where

$$f(\mathbf{n}) = \begin{cases} \exp\left(-\cos^{-1}(\mathbf{n}\cdot\mathbf{n}_{t+1})\,\frac{1}{2s^2}\right) & \text{if } \cos(\mathbf{n}\cdot\mathbf{n}_{t+1}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and s a constant blurring factor. In practice, our system initializes the light map using the solution of the linear system of equations and then updates it for each new frame.

4.3 Offline Estimation of Light Distribution for Rendering Specular Effects and Casting Shadows

While the previous method yields satisfying results for fast online pre-visualization, its accuracy suffers from two numerical problems. First, it does not minimize a physical error since the problem is intrinsincally non-linear. Second, it solves simultaneously for camera-related parameters, the gains and the biases, and lighting-related parameters, the irradiances. As is the case for geometric camera calibration, performing non-linear optimization and dissociating the estimation of the internal parameters, here the gains and biases, from that of the external ones, here the irradiances, yields more reliable results.

Therefore, we developed an offline approach to doing so. It is computationally more expensive than the online technique of Section 4.2 but produces a more sophisticated illumination model that allows for specular highlights, cast shadows, and changing the materials of the virtual objects.

We model the lighting environment as a regularly sampled dome of lights of varying power, such as the one depicted by Fig. 6(b). We begin by estimating the gain and bias of each camera in a way that is independent from lighting effects and normalizing the pixel intensities. We then apply a regularized deconvolution algorithm on the observed pixel intensities within the calibration object to assign to each individual light the power that best explains the observations.

4.3.1 Relative Cameras Responses

To estimate the gains and biases, we consider pairs of views taken at the same time by two different cameras. As we will see, this removes the need from separately computing the surface albedo a_π and the irradiance x_π, thus avoiding an additional source of noise. Let D be the set of triplets (c,d,t) such that both cameras c and d see the calibration object at time t. Assuming a Lambertian behavior of the calibration pattern, we can compute our gains and biases as

$$\underset{g_1 \ldots g_C, b_1 \ldots b_C}{\operatorname{argmin}} \sum_{(c,d,t)\in D} \sum_{\pi} \left(\frac{I(c,\pi,t) - b_c}{g_c} - \frac{I(d,\pi,t) - b_d}{g_d} \right)^2 . \quad (8)$$

We can now define a normalized image $\bar{I}$ from the image taken by camera c as

$$\bar{I}(c,\pi,t) = \frac{I(c,\pi,t) - b_c}{g_c} , \quad (9)$$

and express the deconvolution in terms of these normalized images.

4.3.2 Deconvolution

From Eqs. 5,6, and 9, the normalized intensity at time t of a patch π with normal $\mathbf{n}$ can be written as

$$\bar{I}(c,\pi,t) = a_\pi \sum_{l=0}^{L} \max(\mathbf{n}_t \cdot \mathbf{d}_l, 0)\Omega_l , \quad (10)$$

where L is the number of sources used to represent the lighting environment and $\mathbf{d}_l$ the direction of source l with power Ω_l. Our objective is to recover the powers Ω_l of each source so as to minimize

$$\varepsilon_{obs}(\Omega_l) = \sum_{c,\pi,t} \left[\bar{I}(c,\pi,t) - a_\pi \sum_{l=0}^{L} \max(\mathbf{n}_t \cdot \mathbf{d}_l, 0)\Omega_l \right]^2 , \quad (11)$$

which represents the sum of the squares of the differences between the intensities predicted by the lighting model and the observed ones. Unfortunately, since the Lambertian model acts as a low-pass

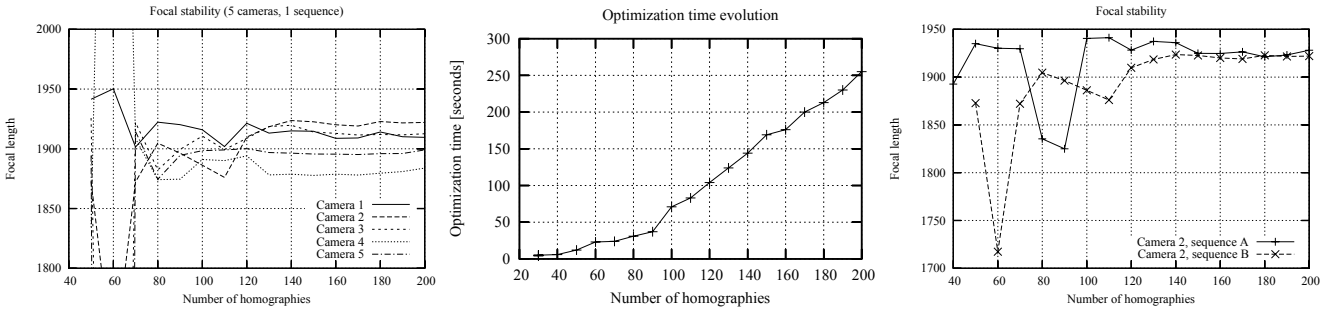


Figure 7: Geometric calibration of 5 cameras. (a) Focal lengths estimated for each camera as a function of the total number of homographies retained. The values stabilize once enough homographies are used. (b) The computational cost grows linearly as a function of the number of homographies. (c) Focal length of the second of 5 calibrated cameras computed independently using two different sets of sequences. Once enough homographies are used, the estimates become very close.

filter on the lighting, this is an ill-posed deconvolution problem that has many potential solutions. To obtain a realistic one, we must first constrain the Ω_l to be positive as not doing so would allow the existence of lights with negative power. To this end, we optimize with respect to the square roots $\Omega'_l = \sqrt{\Omega_l}$, instead of the Ω_l themselves. We then define two regularization terms to be added to $\varepsilon_{obs}(\Omega_l)$. The first one is a smoothness term similar to the one introduced in [14] that we write as

$$\varepsilon_{adj}(\Omega'_l) = \sum_{i,j \text{ adjacent}} \left[(\Omega'_i)^2 - (\Omega'_j)^2 \right]^2 . \quad (12)$$

Minimizing it forces adjacent sources to be of relatively similar powers. The second is designed to encourage the good localization of narrow light sources and is written as

$$\varepsilon_{stick}(\Omega'_l) = \sum_{l}^{L} \left[\left(\Omega'_{0,l}\right)^2 - (\Omega'_l)^2 \right]^2 . \quad (13)$$

In practice, we therefore solve

$$\underset{\Omega'_l}{\text{argmin}} \left(\varepsilon_{obs}(\Omega'_l) + \alpha \varepsilon_{adj}(\Omega'_l) + \beta \varepsilon_{stick}(\Omega'_l) \right) , \quad (14)$$

where α and β are regularization constants. To this end, we use a standard Levenberg-Marquardt least-squares solver but define a specific optimization schedule that initially favors a relatively small subset of light sources and then progressively allows more sources to contribute to the lighting. At the first iteration, the initial power of the sources $\Omega'_{0,l}$ are set to zero and the parameter β is given a high value. This results in an initially small set of sources with significant amounts of power while the majority of the remaining ones are still turned off. The next iterations start from the previous power estimation and the β parameter is progressively decreased to allow the recovery of more wide-spread sources. This results is a lighting sphere with smooth but well-defined clusters instead of a completely smoothed out distribution.

The knowledge of the source positions allows for realistically shaded virtual objects. Realistic cast shadows and specularities can be computed using a Phong shading model. As shown in the next section, the conjunction of tracking and lighting distribution estimation allows us to render convincing augmented images.

5 Results

In this section we first evaluate the accuracy and reliability of the geometric camera calibration of Section 3. We then show that using both the corresponding geometric camera parameters and the photometric ones returned by the procedure of Section 4 lets us augment the scene with objects that are correctly registered *and* lighted.

	Cam 1	Cam 2	Cam 3	Cam 4	Cam 5
τf	-0.16%	0.32%	0.27%	1.12%	0.56%
f	-0.08%	0.33%	0.19%	1.22%	0.53%
u_0	5.57%	6.49%	9.81%	5.85%	6.00%
v_0	0.11%	0.76%	3.15%	-11.7%	-3.56%

Table 1: Recovery of the intrinsic parameters that appear in the $\mathbf{K}_c$ matrix of Eq. 2. We express the differences between those estimated using the first and the second set of sequences as a percentage. In both cases, we used 200 homographies. The percentages for the focal lengths are very small. Those corresponding to the principal points are a bit larger, which is not surprising given the fact that they are known to have much less influence on the projection matrix.

5.1 Accuracy of the Geometric Calibration Parameters

To test the accuracy of the camera parameters that our system recovers, we first trained a classifier to recognize the interest points of the pattern of Fig. 3(a), as discussed in Section 3.1.1. We then used a 5 cameras setup to record two different sets of video sequences by waving the pattern in front of the cameras. Finally, we performed the calibration independently for each set.

Fig. 7(a) depicts the focal lengths recovered using the first set of sequences. They are shown as a function of the total number of homographies retained to perform the computation. When this number climbs above 140, the estimates become quite stable. It therefore does not make sense to use many more since the computational cost increases almost linearly with this number, as shown in Fig. 7(b).

In Fig. 7(c) and Table 1 we compare the results obtained independently using the two sets of video sequences. In Fig. 7(c), we superpose the focal length estimates, again drawn as a function of the total number of homographies retained. As before, once we use more than 140, the two estimates become very close. This is a very good indication that they are accurate since they were computed independently. As shown by Table 1, this is true not only for the focal lengths but also for the principal point locations.

We have not obtained ground truth for the external camera parameters. However, as evidenced by the supplementary videos, the virtual object appears to be very stable with respect to the calibration pattern, which would not be the case if they were poorly recovered.

5.2 Adding and Relighting Virtual Objects

We used a three camera set-up similar to the one depicted by Fig. 1(a) to produce the augmented images of Fig. 8. In the first row, we use only the geometric calibration parameters to draw the virtual teapot at the right place and use a randomly selected point light source to relight it. Even though the teapot is correctly registered, the result is unconvincing because the shading patterns do not match those of the other real objects present in the scene. In the middle row, we use the output of the online photometric calibration procedure of Section 4.2 to relight the object. The result is much improved but highlights and shadows are still missing. As shown in the bottom row of the figure, using the output of the more sophisticated offline calibration procedure of Section 4.3 solves both problems. Both highlights and shadows now appear at the right places, thus significantly increasing the realism. We supply the corresponding videos as supplementary material.

Fig. 9 showcases the flexibility that our multi-camera system provides. In two of the three images, the calibration pattern is not visible in the view we are trying to augment and a monocular approach that relies on detecting it would fail. However, because it is seen by another camera and because the relative positions of the camera with respect to each other have been computed, we can nevertheless draw it at the right location, as evidenced by the fact that the real box occludes it correctly.

Our system can use any sufficiently textured pattern for calibration purposes. In Fig. 10, we demonstrate this by using the piece of cardboard of Fig. 3(b) to calibrate and augment the scene. Note that the virtual teapot casts a realistic shadow on the real cardboard, thereby indicating that both the geometric and photometric calibration are correct. We have also successfully used the book of Fig. 3(c) for this purpose.

6 Conclusion

We have presented a system for simultaneous geometric and photometric calibration of multiple cameras that is extremely easy to use. At run-time, the only manual intervention required involves waving a planar calibration pattern in front of the cameras. Furthermore, it imposes very few constraints on the relative positions and orientations of the cameras and can handle complex real-world illuminations.

A natural extension would be to correct for distorion [21] and vignetting. However, even without it, our system yields both accurate camera parameters that let us add virtual 3D objects at the appropriate locations and a lighting model that allows us to relight them convincingly. Since these are the two ingredients that are required by Augmented Reality applications, we believe that our approach to calibration is ideally suited for this purpose. We urge interested readers to download the source code to try it for themselves [1].

Appendix

Internal Parameters from a Set of Homographies

The computation of internal parameters from a set of homographies is quite standard [17, 21], but we give it here for the sake of completeness. First, we write the matrix of internal parameters as

$$\mathbf{K} = \begin{bmatrix} \tau f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where f represents the focal length, τ the aspect ratio, and $(u_0, v_0)^\top$ the principal point. We want to estimate $\mathbf{K}$ from a set of homographies that map points on a planar object to points on captured images for different object positions.

We associate to each camera a projection matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}], \tag{15}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ a translation vector. Without loss of generality, we can choose $Z = 0$ as the plane of our calibration pattern. The relation between $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{t}$ and the homography $\mathbf{H}$ that maps a point $\mathbf{M} = [X, Y, 0]^\top$ of this plane to its corresponding 2D point $\mathbf{m} = [u, v]^\top$ under perspective projection can be written as

$$\begin{bmatrix} \mathbf{m} \\ 1 \end{bmatrix} \propto \mathbf{P} \begin{bmatrix} \mathbf{M} \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{16}$$

where $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ respectively are the first, second and third column of the rotation matrix $\mathbf{R}$, and the symbol $\propto$ means "equal up to a scale factor". Identifying the terms of Eq. 1 and Eq. 15 yields $\mathbf{H} \propto \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$. For convenience, we introduce a 3×3 matrix $\mathbf{T}$ with

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & \mathbf{t}'_1 \\ 0 & 1 & \mathbf{t}'_2 \\ 0 & 0 & \mathbf{t}'_3 \end{bmatrix},$$

where $\mathbf{t}' = \mathbf{R}^{-1}\mathbf{t} = \mathbf{R}^\top\mathbf{t}$ that lets us write

$$\mathbf{H} \propto \mathbf{KRT}. \tag{17}$$

To find the relations between the coefficients of $\mathbf{H}$ and the internal parameters, let us now compute the product $\mathbf{H}^\top\omega\mathbf{H}$ where ω is the matrix $(\mathbf{KK}^\top)^{-1}$, also known as the image of the absolute conic. We have

$$\mathbf{H}^\top\omega\mathbf{H} = \mathbf{H}^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{H} \propto (\mathbf{KRT})^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}(\mathbf{KRT})$$

and

$$(\mathbf{KRT})^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}(\mathbf{KRT}) = \mathbf{T}^\top\mathbf{R}^\top\mathbf{K}^\top\mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{KRT} = \mathbf{T}^\top\mathbf{T} = \begin{bmatrix} 1 & 0 & -\mathbf{t}'_1 \\ 0 & 1 & -\mathbf{t}'_2 \\ -\mathbf{t}'_1 & -\mathbf{t}'_2 & \|\mathbf{t}'\|^2 \end{bmatrix}.$$

By considering the expressions of the elements of the 2×2 submatrix on the top-left of the $\mathbf{T}^\top\mathbf{T}$ matrix, we obtain the two following equations:

$$\begin{cases} \mathbf{h}_1^\top\omega\mathbf{h}_1 - \mathbf{h}_2^\top\omega\mathbf{h}_2 = 0 \\ \mathbf{h}_1^\top\omega\mathbf{h}_2 = 0 \end{cases}, \tag{18}$$

where $\mathbf{h}_i$ represents the column i of $\mathbf{H}$. It follows that:

$$\omega \propto \begin{bmatrix} 1 & 0 & -u_0 \\ 0 & \tau^2 & -\tau^2 v_0 \\ -u_0 & -\tau^2 v_0 & \tau^2 f^2 + u_0^2 + \tau^2 v_0^2 \end{bmatrix}. \tag{19}$$

By rearranging the terms of Eq. 18 and Eq. 19, we obtain the following linear system in some of the coefficients of ω:

$$\mathbf{AW} = \mathbf{h} \tag{20}$$

where

$$\mathbf{A} = \begin{bmatrix} 2(h_{11}h_{31} - h_{12}h_{32}) & h_{21}^2 - h_{22}^2 & 2(h_{21}h_{31} - h_{22}h_{32}) & h_{31}^2 - h_{32}^2 \\ h_{11}h_{32} + h_{12}h_{31} & h_{22}h_{21} & h_{32}h_{21} + h_{22}h_{31} & h_{32}h_{31} \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{bmatrix} \quad \text{and} \quad \mathbf{h} = \begin{bmatrix} h_{11}^2 - h_{12}^2 \\ h_{11}h_{12} \end{bmatrix}.$$

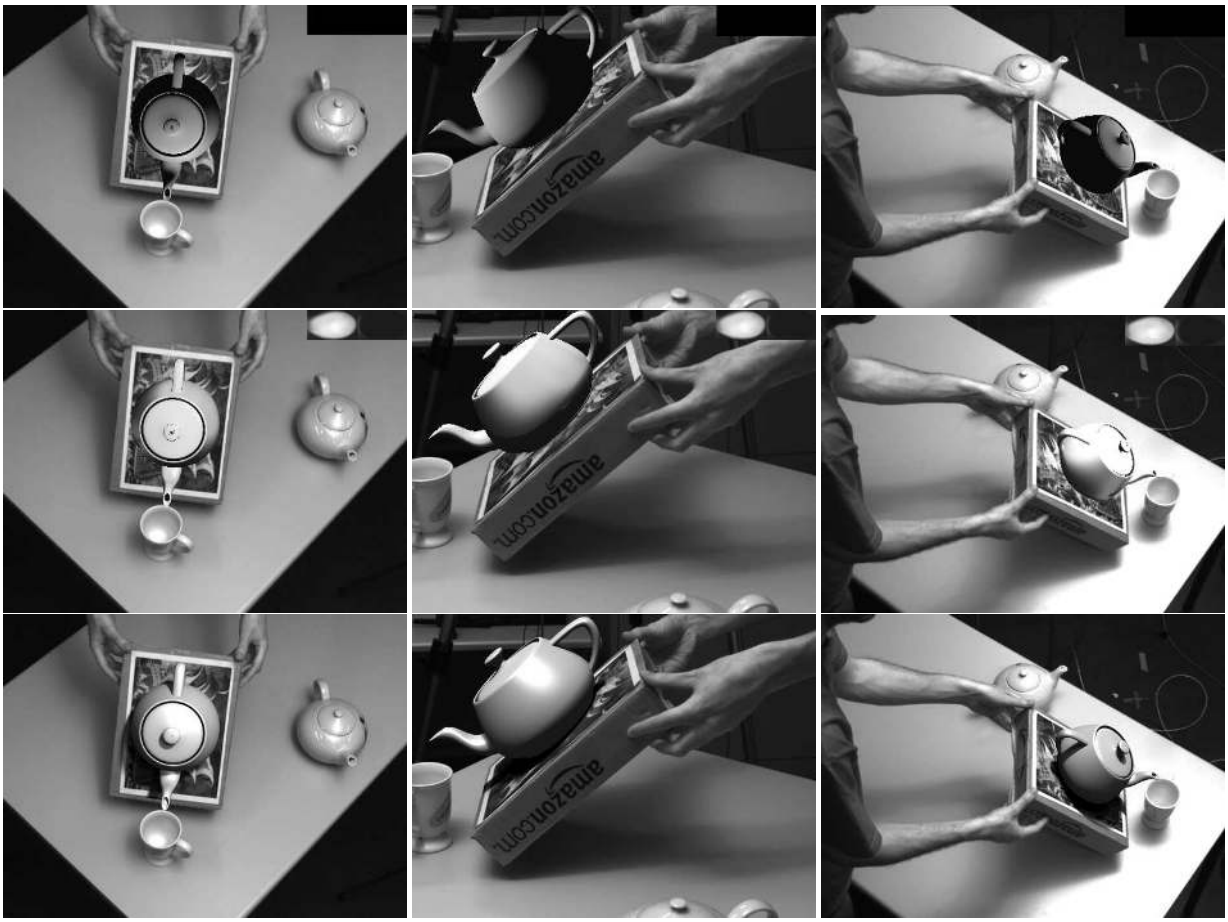


Figure 8: Adding a virtual teapot. **First row**: We use only the geometric calibration data and relight the teapot using a random point light source. The shading patterns do not match those of the real objects. **Second row**: We use the online model of Section 4.2 to relight the teapot. The result is better but the highlights and shadows are still missing. **Third row**: Using the more sophisticated offline technique of Section 4.3 produces realistic highlights and shadows.

Figure 9: Even when the calibration pattern is invisible in the view we are augmenting, the virtual object is accurately registered and correctly occludes the virtual teapot. This is possible because the calibration pattern is seen by another camera whose position and orientation are known.

Each homography yields such a pair of equations. For each camera, this produces an over-constrained system that we solve in the least-squares sense. The internal parameters u_0, v_0, f, and τ can then be estimated from ω_{13}, ω_{22}, ω_{23}, and ω_{33}. In our implementation, they are refined by the final non-linear optimization.

Displacement between a Camera and a Planar Object from a Set of Homographies and the Internal Parameters

Once the internal camera parameters have been recovered from the whole sequencem we can recover the rotation R and the translation t between a particular pose of the planar object and the camera as follows.

Recall from Eq. 17 that $\mathbf{H} \propto \mathbf{KRT}$, where $\mathbf{R}$ and $\mathbf{t}$ are expressed in a coordinate system attached to the planar object. Equivalently, we write:

$$\mathbf{RT} \propto \mathbf{K}^{-1}\mathbf{H} .$$

Since the columns of R should have a norm equal to 1, the scale factor can be retrieved, and a first estimation of these columns is obtained as:

$$\mathbf{r}_1 = \frac{\mathbf{K}^{-1}\mathbf{h}_1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|}, \quad \mathbf{r}_2 = \frac{\mathbf{K}^{-1}\mathbf{h}_2}{\|\mathbf{K}^{-1}\mathbf{h}_2\|}, \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \quad ,$$

where $\times$ denotes the cross-product of two vectors. Because the ho-

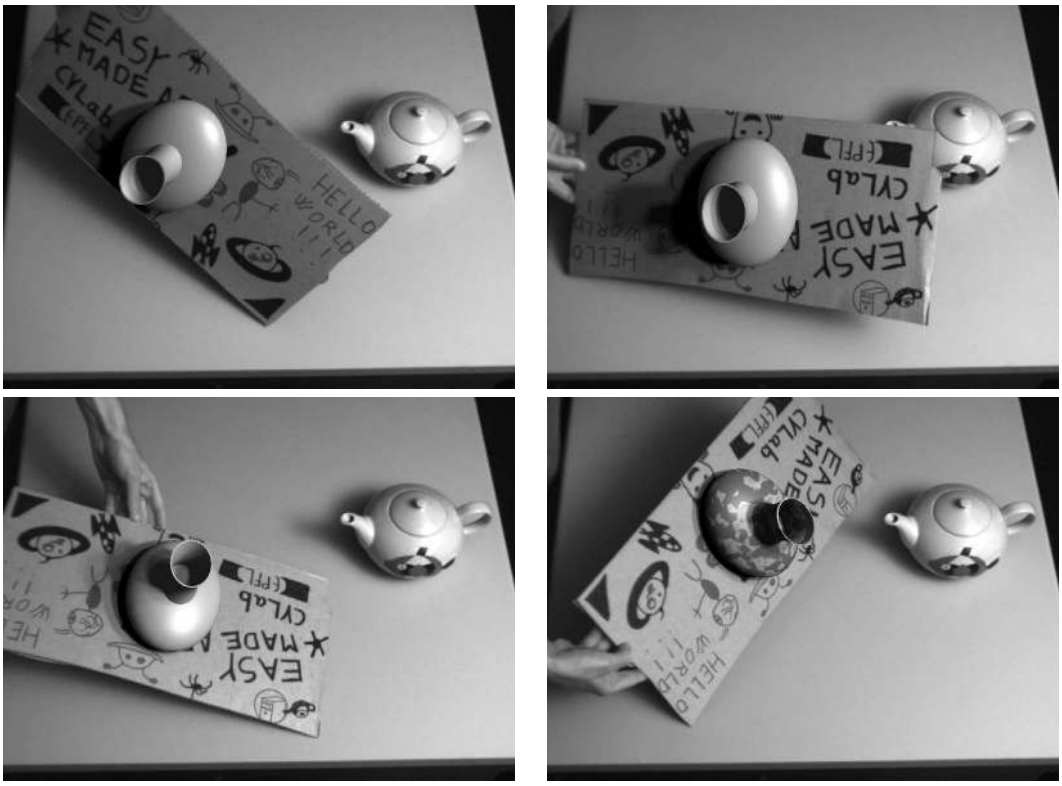


Figure 10: Calibration and augmentation of the piece of cardboard of Fig. 3(b). Note that the virtual vase casts shadows on the real object in all frames and that a specularity moves realistically on its surface. In the bottom right image, we used a texture map to change the albedos of the vase.

mographies are noisy, the resulting rotation matrix is not orthonormal and we correct it using the procedure given in the appendix of [21] which seeks the closest orthonormal matrix in the Frobenius sense using a Singular Value Decomposition. Similarly the translation vector $\mathbf{t}$ can be approximated as:

$$\mathbf{t} = \frac{2\mathbf{K}^{-1}\mathbf{h}_3}{\left\|\mathbf{K}^{-1}\mathbf{h}_1\right\| + \left\|\mathbf{K}^{-1}\mathbf{h}_2\right\|} .$$

REFERENCES

[1] CVLab Software. `http://cvlab.epfl.ch/software/`.

[2] Y. Amit and D. Geman. Shape Quantization and Recognition with Randomized Trees. *Neural Computation*, 9(7):1545–1588, 1997.

[3] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH*, July 1998.

[4] G. Drettakis, L. Robert, and S. Bougnoux. Interactive common illumination for computer augmented reality. In *Eurographics Rendering Workshop*, pages 45–56, June 1997.

[5] M. Fiala and C. Shu. Fully Automatic Camera Calibration Using Self-Identifying Calibration Targets. Technical report, National Research Council Canada, 2005.

[6] S. Gibson and A. Murta. Interactive Rendering with Real-World Illumination. In *Eurographics Workshop on Rendering*, June 2000.

[7] J. Gross and J. Yellen. *Graph theory and its applications*. CRC Press, Inc., Boca Raton, FL, USA, 1999.

[8] C.G. Harris and M.J. Stephens. A Combined Corner and Edge Detector. In *Fourth Alvey Vision Conference, Manchester*, 1988.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[10] M. Kanbara and N. Yokoya. Real-Time Estimation of Light Source Environment for Photorealistic Augmented Reality. In *International Conference on Pattern Recognition*, 2004.

[11] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. In press.

[12] V. Lepetit, P. Lagger, and P. Fua. Randomized Trees for Real-Time Keypoint Recognition. In *Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[13] H.G. Maas. Image sequence based automatic multi-camera system calibration techniques. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(5-6):352–359, December 1999.

[14] S. R. Marschner and D. P. Greenberg. Inverse lighting for photography. In *Proceedings of the Fifth Color Imaging Conference, Society for Imaging Science and Technology*, 1997.

[15] Open Source Computer Vision Library. Intel. http://www.intel.com/technology/computing/opencv/.

[16] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a Radiance Distribution to Superimpose Virtual Objects onto a Real Scene. *IEEE Transactions on Visualization and Computer Graphics*, 1999.

[17] P. Sturm and S. Maybank. On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications. In *Conference on Computer Vision and Pattern Recognition*, pages 432–437, June 1999.

[18] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[19] T. Ueshiba and F. Tomita. Plane-based calibration algorithm for multi-camera systems via factorization of homography matrices. In *International Conference on Computer Vision*, pages 966–973, 2003.

[20] Viconpeak. http://www.vicon.com/.

[21] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 2000.