Harnessing Computer Vision to Project Video on Moving Object Surfaces

Samuel Audet

Master of Engineering

Masatoshi OKUTOMI & Masayuki TANAKA Laboratory

Department of Mechanical and Control Engineering
Graduate School of Science and Engineering
Tokyo Institute of Technology
Toyko, Japan

February 2010

Abstract

Computer vision allows machines to understand their surroundings through video cameras. We can use this understanding to design user interfaces that work with everyday objects. The system can also use video projectors to display feedback directly on these objects. However, the projection can severely alter their appearances to the point where traditional computer vision algorithms fail. We propose an image alignment method that not only can work under these conditions, but that harnesses the knowledge of the displayed content as additional information to help the system align it with object surfaces. We implemented in software an algorithm that successfully executes on planar surfaces and commodity hardware at about two frames per second with subpixel accuracy. We also designed user-friendly methods to perform the required geometric and color calibration. Although slow and limited to planes, we proved the viability of the concept, paving the way for future optimization and generalization.

Acknowledgements

First and foremost, I am much indebted to my advisors Masatoshi OKUTOMI and Masayuki TANAKA for the research opportunities, the trust, and guidance they have given me. Furthermore, based on their recommendations, the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of the Japanese Government awarded me a generously full paid research scholarship, without which this work would not have been possible. I will also be forever thankful to Linda Cooper for teaching me how to properly and clearly write scientific material. Without her wonderful seminars, I fear that I would have never received the necessary knowledge to effectively communicate my findings to others.

Contents

1 Introduction **5**
- 1.1 Literature Review . . . 5
- 1.2 Proposed Method . . . 6
- 1.3 User-Friendly Calibration . . . 7
- 1.4 Direct Image Alignment . . . 9

2 System Model **10**
- 2.1 Simplifying Assumptions . . . 10
- 2.2 Geometric Model . . . 11
- 2.3 Color Model . . . 11
- 2.4 Image Formation Model . . . 12

3 User-Friendly Calibration **14**
- 3.1 Geometric Calibration . . . 14
  - 3.1.1 Detection of Fiducial Markers . . . 14
  - 3.1.2 Calibration Patterns . . . 17
  - 3.1.3 Prewarp of the Projector Pattern . . . 18
  - 3.1.4 Real-Time Algorithm and Issues . . . 19
  - 3.1.5 Calibration Algorithm . . . 22
- 3.2 Color Calibration . . . 23

4 Direct Image Alignment **26**
- 4.1 Cost Function . . . 26
- 4.2 Minimization . . . 27
- 4.3 Initialization . . . 28

5 Results **31**
- 5.1 Geometric Calibration . . . 31
- 5.2 Color Calibration . . . 33
- 5.3 Image Alignment . . . 34

6 Discussion and Conclusion **39**

List of Figures

1.1 Snapshot of our demo video showing the patterns of Figure 5.5, where the system has aligned the projector displayed pattern with the printed one. 6
1.2 Snapshot of our demo video showing the test user calibrating a casually installed projector-camera system. 7

2.1 Sketch of the image formation model. 12

3.1 Sample camera image captured during geometric calibration, where the projector markers are in white, and the printed markers in black. 15
3.2 Two sample fiducial markers with IDs 0 and 1 made from the binary-coded BCH code of ARToolKitPlus. 16
3.3 Sample calibration patterns composed of matrices of markers. 17
3.4 The ideal image that would be created by projecting the pattern from Figure 3.3(b) over Figure 3.3(a) under perfect alignment. 18
3.5 Flowchart illustrating one iteration of the interactive real-time algorithm. 20
3.6 Dataflow diagram for calibrating the camera and the projector. 22
3.7 Lower left area of the detected markers from Figure 3.5. We can observe severe defocusing of projector markers. 23
3.8 Sample camera image captured during color calibration when the projector displays the color blue at maximum intensity. . . 24
3.9 Blank area automatically extracted from Figure 3.8, whose average is used during color calibration. 24
3.10 Samples from the RGB color space of the projector used during color calibration. 25

4.1 Fractal image displayed at the third phase of the initialization. 29

5.1 Predicted and observed intensities of the camera. Predicted values computed as per Equation 2.4 using parameters of Table 5.3. Observed values come from data taken during the color calibration using the sampling pattern discussed in Section 3.2. 33
5.2 The printed triangular fractal pattern along with four markers that we used to obtain accuracy data. 34
5.3 Frames from the marker test video with offline drawn red crosses representing the detected markers and green rectangles denoting the corresponding regions aligned by our program. The rectangle corners match the crosses with subpixel accuracy as shown in the blowups. 35
5.4 Difference in pixels between our results and the centers of the markers. 36
5.5 The images used for our demo video. 37
5.6 Frames from our demo video. We grouped them in pairs of a misaligned image caused by user motion followed by its correction. 38

6.1 Screenshot of the main window of ProCamCalib, where all the settings can be adjusted. They are currently configured for calibrating two cameras and one projector. 40

List of Tables

5.1 Reprojection RMSE, root mean square errors, in pixels after calibrating using our method with either corners or centers. . . 31
5.2 Reprojection RMSE, root mean square errors, in pixels of other methods. For tidiness, we cite only the first author. . . . 32
5.3 Sample output of the color calibration, with bias $\mathbf{b}$ = zero. . . 33
5.4 Average time per iteration of the marker test. 36

Chapter 1

Introduction

Traditional applications of augmented reality superimpose generated images onto the real world through goggles or monitors held between objects of interest and the user. The display must usually follow objects and developers often choose cameras to perform tracking, as computer vision methods are flexible and nonintrusive. Since the augmented objects remain in reality unchanged, we do not need to change the image processing algorithms either. One can directly apply existing computer vision techniques for tracking or other purposes. However, with spatial augmented reality [7], we instead use video projectors to display computer graphics or data onto surfaces, as exemplified in Figure 1.1. In this case, the appearance of the objects may be severely affected, requiring new methods.

1.1 Literature Review

To avoid the difficulty, current methods either exploit information channels that do not overlap with the displayed content or make assumptions that restrict their usefulness. Some use fiducial markers, such as iLamps [28], or a special tracking system not based on visible light, such as Dynamic Shader Lamps [6], but users need to paste specially prepared markers to objects they might want to interact with. Others have designed imperceptible structured light, such as used in the Office of the Future [29]. However, this not only reduces the dynamic range of the projector, but requires a synchronized projector-camera system running at frequencies higher than 60 Hz to avoid flicker. More simply the Perceptive Workbench [24] has adopted near-infrared cameras and uses computer vision to track without interference from the projector, but calibrating projectors and cameras whose light spectra do not overlap can be problematic. Although all the above options

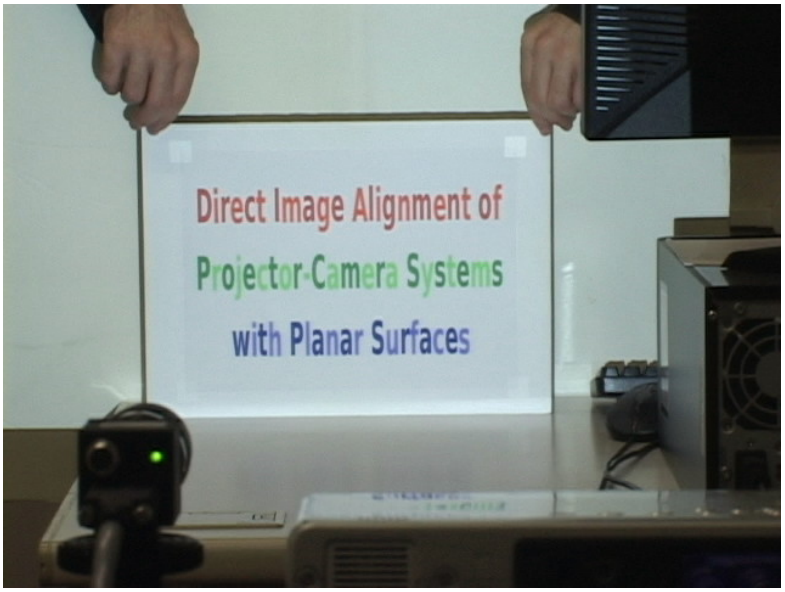


Figure 1.1: Snapshot of our demo video showing the patterns of Figure 5.5, where the system has aligned the projector displayed pattern with the printed one.

work, they cannot directly align displayed content with real world texture. They solely depend on accurate geometric calibration between the sensors, the projectors, and the real world objects. Also, the system does not see the same thing as the user, possibly causing confusion when the algorithm fails. In another direction, Tele-Graffiti [35] features a paper tracking algorithm that detects the orientation of a piece of paper or a clipboard inside the images captured using a normal camera. The authors designed the algorithm robustly enough to work in the one limited case where the edges of the rectangular object differ markedly from the background, but they still consider the displayed content from the projector as unwanted interference.

In a nutshell, the performance of markerless tracking leaves to be desired. If spatial augmented reality is to become the basis for a new paradigm of user interaction, we need more general, robust, and easy-to-use methods.

1.2 Proposed Method

Contrarily to existing approaches that treat the displayed content as interference, we propose to harness its knowledge as additional information that can help the system align it with real world objects. More specifically, we derived a direct image alignment algorithm that takes projector illumination

into account using a generative model. It models how the light coming out of the projector reflects onto a real world object and comes back in the camera.

In theory, the model can work with an uncalibrated system, but a calibrated one can more easily achieve successful alignment. Calibration amounts to finding the internal and external geometric and color parameters of both camera and projector. Although calibration methods exist, they are cumbersome, impractical, and could not realistically become part of an augmented reality system that anyone could use under a large range of casual conditions, such as the one shown in Figure 1.2. For this reason we also designed geometric and color calibration methods, which we briefly introduce next, that users of any background may easily perform.

1.3 User-Friendly Calibration

A lot of methods to geometrically calibrate projector-camera systems, [1, 2, 18, 22, 27, 38] among others, operate in two phases: they first calibrate cameras, then they calibrate projectors. Calibrating cameras in one way and projectors in another automatically doubles the effort. To reduce the amount of work, researchers have proposed methods that integrate both calibrations together. They either exploit structured light or color channels, or both with one-shot structured light.

Figure 1.2: Snapshot of our demo video showing the test user calibrating a casually installed projector-camera system.

In the case of methods using structured light, [12, 16, 23, 25, 38] among others, during the calibration process, the machine may shut the projector off, capture an image, then project structured light, capture more images, and by decoding everything can geometrically relate images together. The ones captured with the projector off are used to calibrate the camera, and the rest, to calibrate the projector. Even though this procedure works, we need to secure physically the calibration target in place, because within a given set of images the machine assumes that the scene is static, often a requirement of structured light. We cannot simply hold a board in our hands in front of the projector-camera system.

To capture two images simultaneously, we can also exploit color channels, [1, 2, 16, 17, 38] among others, usually the red and blue ones. With properly designed complementary colors for the physical and projected patterns, we can recover them separately, *e.g.*: White and cyan squares show clearly in the red channel of a camera, but should hopefully not appear in the blue channel. While this obviously does not work for grayscale projectors, cameras, and printers, even with color ones, this approach presents two problems. First, depending on the properties of the color filters and inks, the blue channel of the projector might still interfere with the red channel of the camera, and vice versa. For example, results from Chen *et al.* [10] show a 4% overlap between the red channel of their camera and the blue channel of their projector, and the same the other way around. Second, many cameras use only one sensor with an appropriate array of color filters, such that the resolution of the blue and red channel is half of the resolution of the whole sensor. Both of these issues may hurt accuracy, although current results are not conclusive, but more importantly requiring color complicates the execution of calibration.

As for color calibration, current methods [9, 10, 11, 21, 32] require either control over the shutter speed of the camera, more than one projector, or an elaborate 3D contraption, but we wanted a method that works without camera control and with only one projector and a planar surface.

For all the above reasons, we propose instead a user-friendly method to perform full geometric and color calibration of a projector-camera system. It is based on fiducial markers typically used for augmented reality applications. Fiala and Shu [14] proposed such a method for camera-only systems. We extend it to projector-camera systems plus color calibration, where half of the markers are physically printed, and half are projected using the projector. Each marker on its own carries information and can easily be identified. As we describe in this document, this allows the machine to prewarp markers that are projected, in a way that does not interfere with printed markers. Moreover, markers do not need color, and unlike structured light users can hold the calibration board in their hands. The user simply has to wave the

calibration board, ideally at angles of approximately 45° with respect to the image plane as recommended by Zhang [39], and the machine automatically captures and saves about ten good images, with which it computes calibration parameters for both the camera and the projector. The machine also uses the markers to detect blank regions on the calibration board, which it takes as reference white to calibrate the colors. Color calibration is relatively simpler since we found that a strictly linear model to match colors between the camera and the projector works well.

Our results show that users could perform full geometric and color calibration in less than one minute, a use case that we consider user-friendly. Calibration results are also accurate, with less than one pixel of geometric error and, for color, a correlation coefficient R^2 of virtually one.

1.4 Direct Image Alignment

Once fully calibrated, image alignment becomes more robust. Nevertheless, even given calibration, it remains a challenging task. As a first step, we decided to make a few simplifying assumptions. Most importantly, the object must be planar and feature diffuse reflectance properties with no specular reflections. Still, thanks to the inherent robustness of direct alignment methods, we found that the algorithm can cope with large amounts of noise. To provide some results, we implemented in software our method for planar surfaces.

At this time, the program runs only at about two frames per second on commodity hardware, and this restricts the speed at which a user can move the object, but it achieves subpixel accuracy. Future research will revolve around optimization and generalization.

In the following sections, we first describe in more details the system model and its simplifying assumptions, the required geometric and color calibration procedures, including the detection of fiducial markers, the calibration patterns used, the prewarp of the projector pattern, and the real-time algorithm, and finally the direct image alignment algorithm itself, which minimizes a cost function that requires initialization when presented with a new planar surface. To simplify the explanations, we describe a system with only one camera and one projector, yet it can easily be extended to any number of them.

Chapter 2

System Model

A projector-camera system can be treated as a stereo pair, and multiple view geometry [19] still applies as is. However, unlike traditional stereo systems made from two or more similar camera devices, color information does not transfer as easily from a projector to a camera. We need a more sophisticated color model. In this section, before describing the geometric and color models, to simplify them, we start by enumerating assumptions we found useful.

2.1 Simplifying Assumptions

We first assume that the projector and camera are fixed together, like a typical stereo camera, implying that the strong epipolar constraint applies. The system may nonetheless move with respect to the scene or vice versa, the problem remains unchanged. Secondly, the color responses of both camera and projector must be linear with respect to the light intensity. We consider this to be reasonable for two reasons. On one hand, this is typically the case of CCD or CMOS sensors. On the other hand, most devices today follow the sRGB color space standard [20] that features a well defined color response curve approximating a transfer function with a gamma of 2.2, from which the intensities can be linearized. We expect any deviations to be engulfed in the inaccuracies introduced by the unknown light sources and reflectance properties of the surface material, which brings us to the third assumption. The scene consists of a planar object with a matte surface exhibiting diffuse reflectance, where the radiance changes smoothly and constantly as the angle and distance vary. In particular, specularities are not modeled. Lastly, all light shining on the plane comes from point sources at infinity, such that no local shadows or highlights appear. Only global changes in illumination are observed, which we refer to as *ambient light*.

2.2 Geometric Model

To geometrically model both camera and projector devices, we use the common pinhole camera model [19]. Although originally designed for cameras, it also applies to projectors. It maps a 3D point $\mathbf{x}_3$ of the surface to

$$\mathbf{x}_2 = \mathrm{K}_{3\times3} \left(\mathrm{R}_{3\times3}\mathbf{x}_3 + \mathbf{t}_3 \right), \tag{2.1}$$

a 2D point on the image plane of the device, where K, the camera matrix, contains the internal parameters or intrinsics, and where R and $\mathbf{t}$, the external parameters or extrinsics, model the orientation and position in 3D space of the device. It follows that the projection onto the image planes of a camera placed at the origin and of a calibrated projector can be respectively written

$$\mathbf{x}_\mathrm{c} = \mathrm{K}_\mathrm{c} \left(\mathrm{I} \ \mathbf{x}_3 + \mathbf{0} \right), \tag{2.2}$$

$$\mathbf{x}_\mathrm{p} = \mathrm{K}_\mathrm{p} \left(\mathrm{R}_\mathrm{p}\mathbf{x}_3 + \mathbf{t}_\mathrm{p} \right). \tag{2.3}$$

To avoid confusion, we refer to the matrix K as the *camera matrix* even in the case of a projector. Further, even though equations dealing with homogeneous coordinates are only equal up to an arbitrary scaling factor, we use the equal sign for convenience and clarity.

2.3 Color Model

While the geometric model of a device stands independently on its own, for the color model, we decided instead for simplicity to model only the relationship between devices. Specifically, if a projector pixel shines the color $\mathbf{p}_\mathrm{p}$ on a surface point of *reflectance* $\mathbf{p}_\mathrm{s}$, we expect the corresponding camera pixel to observe the color

$$\mathbf{p}_\mathrm{c} = \mathbf{p}_\mathrm{s} \times \left[\, g\mathrm{X}_{3\times3}\mathbf{p}_\mathrm{p} + \mathbf{a} \, \right] + \mathbf{b}, \tag{2.4}$$

where g is the *gain* of the projector light, which we assume varies smoothly and constantly with the distance and angle to the surface; X is the *color mixing matrix* as defined by Chen *et al.* [10], also called the "projector-camera coupling matrix" by Caspi *et al.* [9]; $\mathbf{a}$, the ambient light; and $\mathbf{b}$, the noise bias of the camera. All vectors are three-vectors in the RGB color space, and their multiplication is done elementwise. We derived the model directly from the bidirectional reflectance distribution function (BRDF) of a flat matte surface, which Johnson and Fuchs [21] have shown to be valid in the case of projector-camera systems. Although the equation explicitly models the relation between only one camera and one projector, as long as one identifies the reference device, the model can adapt to any number of them.

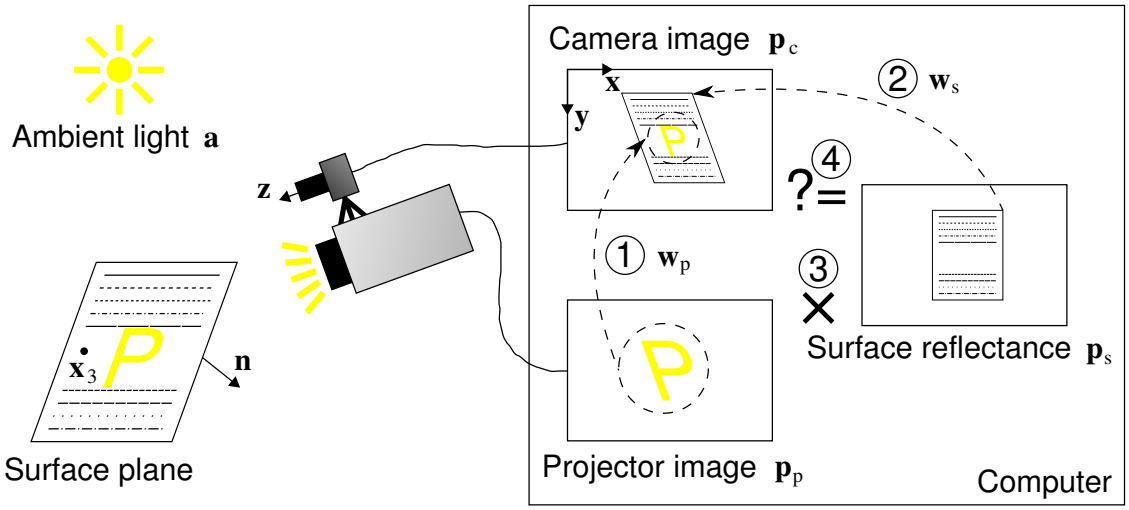


Figure 2.1: Sketch of the image formation model.

2.4 Image Formation Model

Using the geometric and color models, we can simulate how an image forms on the sensor of the camera. The explanation that follows is also summarized in Figure 2.1. Assuming the geometric parameters $\mathbf{n}$ of the plane are known and given a 3D point $\mathbf{x}_3$ on the surface plane such that $\mathbf{n}^T\mathbf{x}_3 + 1 = 0$, it follows that the 2D point $\mathbf{x}_p$ from Equation 2.3 can also be expressed as

$$\begin{aligned} \mathbf{x}_p &= K_p\ (\ R_p\mathbf{x}_3 + \mathbf{t}_p\ ) \\ &= K_p\ (\ R_p\mathbf{x}_3 - \mathbf{t}_p\mathbf{n}^T\mathbf{x}_3\ )\ \text{ since }\ \mathbf{n}^T\mathbf{x}_3 = \text{-}1 \\ &= K_p\ (\ R_p - \mathbf{t}_p\mathbf{n}^T\ )\ \mathbf{x}_3 \\ &= K_p\ (\ R_p - \mathbf{t}_p\mathbf{n}^T\ )\ K_c^{-1}\ \mathbf{x}_c \\ &= \quad H_{pc} \quad\quad \mathbf{x}_c, \end{aligned} \tag{2.5}$$

where the before last substitution comes from the fact that homogeneous vector $\mathbf{x}_c$ from Equation 2.2 can be arbitrarily scaled to fit in. This shows that a camera pixel $\mathbf{x}_c$ is transformed by a homography H_{pc} into the projector pixel $\mathbf{x}_p$. From this we define the *warping function* from the camera image plane to the one of the projector:

$$\mathbf{w}_p(\mathbf{x}_c) \equiv H_{pc}\ \mathbf{x}_c. \tag{2.6}$$

This corresponds to step 1 in Figure 2.1.

Similarly, one can map a camera image point $\mathbf{x}_c$ onto a point $\mathbf{x}_s$ of the image of the surface plane, assumed to have been initialized at a prior moment using the same camera. The camera motion R_s, $\mathbf{t}_s$ required to return to the prior orientation and position can be seen as a second camera (one can think of it as the "surface camera") with the same set of internal parameters, but with different external parameters, as follows:

$$\begin{aligned}
\mathbf{x}_s &= K_c\ (\ R_s\mathbf{x}_3 + \mathbf{t}_s\ ) \\
&= K_c\ (\ R_s\mathbf{x}_3 - \mathbf{t}_s\mathbf{n}^T\mathbf{x}_3\ )\ \ \text{since}\ \ \mathbf{n}^T\mathbf{x}_3 = \text{-1} \\
&= K_c\ (\ R_s - \mathbf{t}_s\mathbf{n}^T\ )\ \mathbf{x}_3 \\
&= K_c\ (\ R_s - \mathbf{t}_s\mathbf{n}^T\ )\ K_c^{-1}\ \mathbf{x}_c \\
&= \qquad H_{sc} \qquad\quad \mathbf{x}_c,
\end{aligned} \tag{2.7}$$

which again becomes a homography H_{sc} that this time maps current camera points $\mathbf{x}_c$ to surface points $\mathbf{x}_s$, from which we define another warping function:

$$\mathbf{w}_s(\mathbf{x}_c) \equiv H_{sc}\ \mathbf{x}_c. \tag{2.8}$$

This corresponds to step 2 in Figure 2.1.

Finally, plugging these coordinates into the color model of Equation 2.4 by considering the pixel colors $\mathbf{p}_c$, $\mathbf{p}_s$, and $\mathbf{p}_p$ as functions over the images, we expect the pixel color at point $\mathbf{x}_c$ of the camera to be

$$\mathbf{p}_c(\mathbf{x}_c) = \mathbf{p}_s(\mathbf{w}_s(\mathbf{x}_c)) \times [\ g\mathrm{X}\mathbf{p}_p(\mathbf{w}_p(\mathbf{x}_c)) + \mathbf{a}\ ] + \mathbf{b}. \tag{2.9}$$

This corresponds to the final third and fourth steps in Figure 2.1.

Chapter 3

User-Friendly Calibration

Before we may use them, these models require a number of parameters to be known, which we can find via calibration. We proceed in a manner similar to current methods for both geometric parameters (K_c, K_p, R_p, and $\mathbf{t}_p$) [3, 39] and color parameters (X and $\mathbf{b}$) [9, 10], but we assume linearized color responses from the start. We first explain geometric calibration, since it can be carried out without color information, while the latter requires prior geometric knowledge.

3.1 Geometric Calibration

To geometrically calibrate the system, we use as is our previously proposed method [3]. It is based on fiducial markers typically used for augmented reality applications. Half of the markers are physically printed in black, and half are displayed in white using the projector, as shown in Figure 3.1. Each marker on its own carries information and can easily be identified. As described below, this allows the machine to prewarp projector markers in a way that does not interfere with printed markers. The user simply has to wave the calibration board in front of the projector and camera, and the machine automatically captures and saves about ten good images, with which it computes calibration parameters for both the camera and the projector via Zhang's method [39]. Also, markers do not need color, so one can calibrate the geometry before the colors.

3.1.1 Detection of Fiducial Markers

As fiducial markers, we opted for the BCH code provided by ARToolKitPlus [37]. Two examples with IDs of 0 and 1 are shown in Figure 3.2. There are 4096

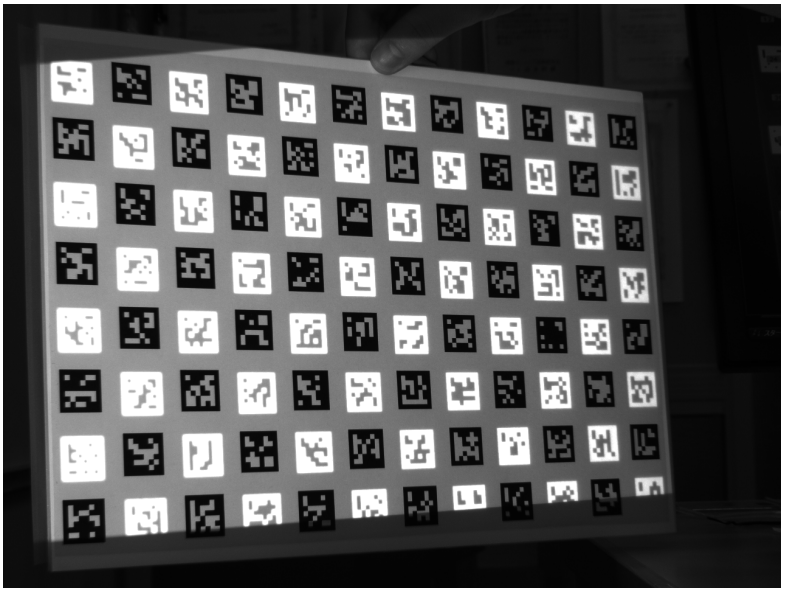
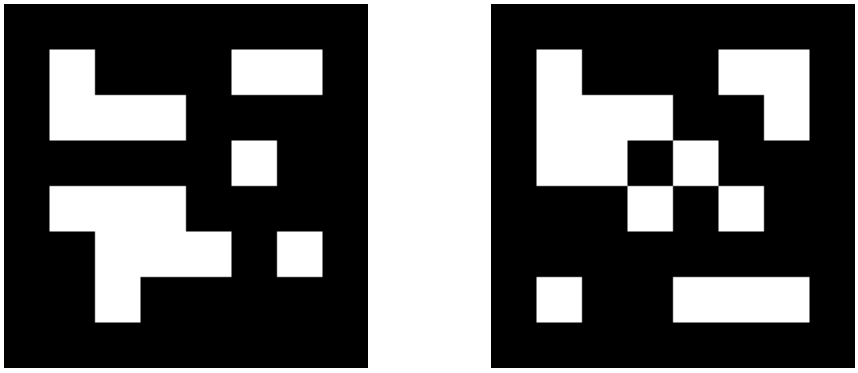


Figure 3.1: Sample camera image captured during geometric calibration, where the projector markers are in white, and the printed markers in black.

such unique markers, arranged in squares of 8×8 black or white subsquares composing the black border and 36 interior bits, for only 12 bits of data, thus featuring strong error correction capabilities. Once printed or projected on a sheet of paper and imaged sufficiently large by a camera, ARToolKitPlus can detect these markers. Basically, the software first binarizes the image at an appropriate threshold, then analyzes contours to find shapes close to a quadrangle, estimates the four corners, and warps the shapes into squares, thus removing projective distortion from the camera. Lastly, by decoding the BCH code inside the squares, it recovers the IDs of the markers. In our case, even if a detected quadrangle is not a marker, since we use a small amount of markers and thanks to error correction, the decoding would most likely simply fail. Fiala [13] provides a more detailed analysis for markers used by ARTag, which are similar to the BCH code used by newer versions of ARToolKitPlus.

While ARToolKitPlus provides most of the desired functionality, we had to add two more features: adaptive thresholding and subpixel corner extraction. The former finds an optimal threshold for binarization even if the level of brightness varies across the image. The latter refines the location of the detected quadrangles, in the original grayscale image.

Adaptive thresholding adjusts the threshold depending on the local neighborhood of each pixel. We adopted Sauvola and Pietikäinen's method [30],



(a) ID = 0. (b) ID = 1.

Figure 3.2: Two sample fiducial markers with IDs 0 and 1 made from the binary-coded BCH code of ARToolKitPlus.

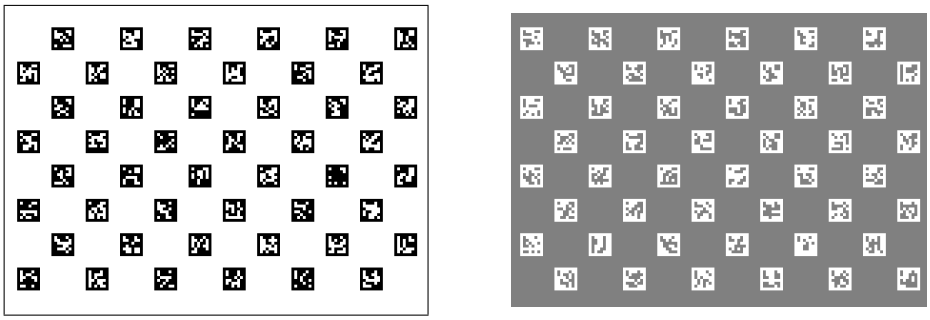
which sets the threshold for each pixel (x, y) to

$$t(x,y) = m(x,y) \times \left[1 + k\left(\frac{s(x,y)}{128} - 1\right)\right] \ , \tag{3.1}$$

where k is a parameter close to zero, usually $[0.0, 0.5]$, and where $m(x, y)$ and $s(x, y)$ are the local mean and the local standard deviation, which are computed within a neighborhood of (x, y). In our case, we considered an adaptive neighborhood size, instead of a constant one. If the variance is very low, then the intuition is that the window of the neighborhood is too small. We need to make it larger to compute a good threshold. Inversely, if the variance is very high, then it is too large. We are no longer analyzing local statistics and the threshold would not be optimal. For each pixel, we thus need to adapt the window size dynamically. We chose to do a binary search to find the size where the standard deviation $s(x, y)$ drops below a certain threshold. Assuming a grayscale image with pixel values in the range $[0, 255]$, since the maximum standard deviation is a bit less than 128 and the minimum is 0, we found 64 to be an appropriate threshold. We implemented an efficient algorithm based on integral images that runs in $O(N^2 \log N)$ time for an $N \times N$ image.

For subpixel corner extraction, we were able to use directly the code provided in OpenCV [8]. The algorithm works by first considering a small window (*e.g.*, 11×11) around the pixel of interest in the original grayscale image. It then assumes that the following holds for all subpixels $\mathbf{x}$ in the window:

$$(\mathbf{c} - \mathbf{x}) \cdot \nabla_{\mathbf{x}} = 0 \ , \tag{3.2}$$



(a) Printed pattern. (b) Projector pattern.

Figure 3.3: Sample calibration patterns composed of matrices of markers.

where $\mathbf{c}$ is the exact subpixel corner location and $\nabla_{\mathbf{x}}$ is the gradient at $\mathbf{x}$. This is true if either $\nabla_{\mathbf{x}}$ equals $\mathbf{0}$ or is orthogonal to $(\mathbf{c} - \mathbf{x})$, which is valid for a corner (but also for a straight edge, so it cannot be used for corner detection). The algorithm finds the minimum of this function, which is not normally zero because of noise and nonlinear distortions.

In this manner, we obtain for each marker an ID and the subpixel coordinates of its four corners.

3.1.2 Calibration Patterns

Even though it is possible to calibrate using only four corners from only one marker, to obtain best calibration accuracy, we should in fact attempt to cover the largest area possible with a great amount of corners [34]. As calibration pattern, we decided to use a matrix of markers, similar to the one that Fiala and Shu [14] used, but where half of the markers are printed and the other half come from the projector, as shown in Figure 3.3. As explained in more details below, the projector markers are in inverted color, since it helps to detect them when projected on top of printed markers. Given the minimum focus distance of our projector, we found that 20×20 mm^2 markers spaced every 30 mm on a B4 size board works well. This gives a total of 12×8 markers, half of which are printed and the rest from the projector.

If one uses a board larger than the field of views, the system would obviously never be able to detect some markers, but as long as it can detect enough of them, there is no problem. In fact, since corners may in that case fully cover the image planes, it should improve accuracy.

In any case, images captured by the camera contain a mix of both markers. For a given image to be considered appropriate for calibration purposes, the patterns need to be aligned well enough such that the markers from the

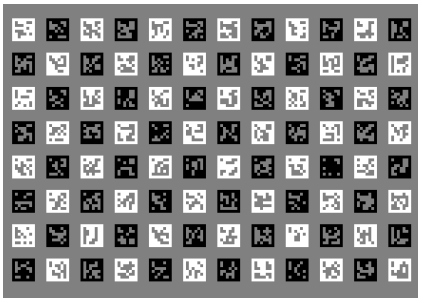


Figure 3.4: The ideal image that would be created by projecting the pattern from Figure 3.3(b) over Figure 3.3(a) under perfect alignment.

projector do not interfere with the printed ones, and vice versa, as shown in Figure 3.4. In the following section, we explain how to accomplish this by prewarping the projector pattern.

3.1.3 Prewarp of the Projector Pattern

For camera-only systems, we can easily accommodate the number of corners shown in Figure 3.4, but this is challenging for projector-camera systems, because we need to image both physical corners and projector corners. Without proper considerations, the pattern from the projector would interfere with the printed one. In this section, we show that, to prevent interference, prewarping the projector markers does not change the calibration problem.

The prewarp transformation that we derive is a homography. Although the patterns contain many markers, we actually only need one to estimate a homography with maximum likelihood under the assumption of Gaussian noise [19]. Still, at least a few markers have to be decodable to compute a meaningful estimate of the error. In cases of large errors, the patterns are not well aligned, and we should not use the estimate for calibration purposes, but we can take it to update the projector prewarp and hope to get a better estimate from the next captured image. For this reason, we still need to find that first homography between the camera and the board.

When a few printed markers are decoded, that homography H_{bc} satisfies

$$\mathbf{x}_{\mathrm{b}} \approx \mathrm{H}_{\mathrm{bc}}\mathbf{x}_{\mathrm{c}} \ , \tag{3.3}$$

where $\mathbf{x}_{\mathrm{b}}$ is a point on the board and $\mathbf{x}_{\mathrm{c}}$, the corresponding point on the camera image plane. H_{bc} is also the initialization information required to calibrate a camera using Zhang's method [39]. For the projector, we have the similar relationship

$$\mathbf{y}_{\mathrm{b}} \approx \mathrm{H}_{\mathrm{bc}}\mathrm{H}_{\mathrm{cp}}\mathbf{y}_{\mathrm{p}} \ , \tag{3.4}$$

where $\mathbf{y}_\mathrm{b}$ is a point on the board; $\mathbf{y}_\mathrm{p}$, the corresponding point on the projector image plane; H_bc, the homography as defined above; and H_cp, the homography transforming points from the image plane of the projector to the one of the camera, which we find using detected projector markers in the camera image. With the combined homography $\mathrm{H}_\mathrm{bc}\mathrm{H}_\mathrm{cp}$, we can calibrate a projector in the same way as a camera. This is how all current projector calibration methods work. However, we can equivalently write

$$\mathbf{y}'_\mathrm{b} \approx \mathrm{H}_\mathrm{bc}\mathrm{H}_\mathrm{cp}\mathbf{y}'_\mathrm{p} \ , \tag{3.5}$$

where $\mathbf{y}'_\mathrm{p} = \mathrm{H}_\mathrm{p}\mathbf{y}_\mathrm{p}$ and H_p is the homography that prewarps our projector image. In other words, we can choose H_p, such that a transformed point $\mathbf{y}'_\mathrm{p}$ on the image plane of the projector appears where we want it on the board, $\mathbf{y}'_\mathrm{b}$, selected as to not interfere with the printed pattern. Because $\mathrm{H}_\mathrm{bc}\mathrm{H}_\mathrm{cp}$ remains untouched, we can say that the calibration method is equivalent.

3.1.4 Real-Time Algorithm and Issues

The sections above fully cover the theoretical aspects, but the machine executes in practice an interactive real-time algorithm. One of its iteration is depicted in Figure 3.5. First, it applies the prewarp of Section 3.1.3 to the projector pattern and sends it to the projector. We assume the user holds at all times the flat board with printed markers in front of the projector-camera system. In this case, projector markers appear along side the printed ones. The system can then continue and attempt to detect all printed and projector markers, one by one as explained in Section 3.1.1. If it considers that the error on the detected corners is low enough (details below) it uses them to update the prewarp homography. Further, if the patterns are aligned well enough (details below) it saves their marker corners. Finally, it loops and expects the user to move the board, unless enough corners have been saved, at which point they are used for calibration.

Although a conceptually simple algorithm, a few thorny technical issues remain. First, if we want the results of the subpixel detector to mean anything, we must make sure the prewarped projector image contains as little aliasing as possible. Next, when projector and printed markers overlap, it is not usually possible to detect them. However, using inverted colors makes them more easily detectable, even when they interfere with each other. Third, even though we can derive a useful update for the prewarp from a rough homography estimation, outliers do not help. Finally, the machine must somehow judge when both patterns are aligned well enough and automatically select images for calibration. In the following subsections we explain in further details and provide practical solutions.

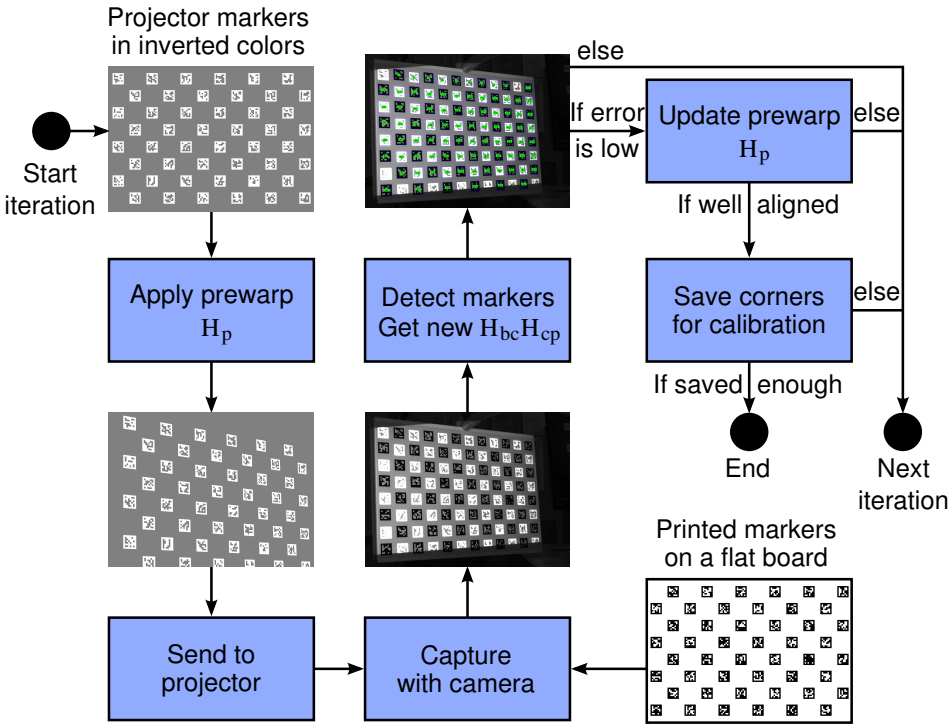


Figure 3.5: Flowchart illustrating one iteration of the interactive real-time algorithm.

Antialiased Projector Images

Typical projector calibration methods do not warp the projector image and do not have to worry about aliasing. In our case, arbitrary warps produce subpixel motion, and aliasing issues must be managed. As antialiasing measure, we opted for simple 4× supersampling [15], which gives good results and executes fast enough in software. Basically, we first create an image buffer that has four times the resolution of the projector. For example, a 4096×3072 image for a typical 1024×768 projector. Next, we draw the markers in the high-resolution buffer using well know drawing and filling algorithms [15] that work on a per-pixel basis. Since we make no attempt at drawing at the subpixel level, this high resolution image contains a lot of aliasing artifacts. However, when smoothed with simple averaging and decimated by a factor of four, the resulting image is subpixel sharp and free from problematic aliasing artifacts. Graphics processing units (GPUs) can also usually perform this operation quickly and efficiently via OpenGL or Direct3D, but the proprietary nature of the actual algorithms they use does not amend itself well to reproducible results. In either case, the system must then send the resulting image to the projector.

Detection of Overlapping Markers

Afterward, if the user holds the calibration board in front of the projector and camera, printed and projector markers usually overlap up to some degree. Because of the limited dynamic range of a typical camera, if we use black markers over a white background only, both black regions from the projector and regions printed in black appear in the same shade of black. Any overlapping markers do not appear as quadrangles anymore, and detection fails. To remedy, we observed that using projector markers with inverted colors (*i.e.*, white markers over a black background) overlapping markers are more robustly detected. This can be explained by the fact that with markers designed this way, the camera automatically images as gray the empty regions of the board, while black printed regions appear black, and white projected regions appear white, naturally providing orthogonal intensity ranges for both types of markers. For this reason, we use inverted colors for projector markers. In addition, the machine can easily adjust the brightness of the markers to match the ambient light and fall within the dynamic range of the camera or alternatively, in dark environments, use the projector as an adjustable source of light, by setting the minimum intensity of pixels.

Rough Homography Estimation

Even if overlapping markers are detected, the accuracy of their corners might obviously suffer, and consequently any homography we might estimate from them. In extreme conditions, markers may also be incorrectly detected. To compensate, we found it works well to reject homographies with large errors (*e.g.*, with an RMSE, root mean square error, greater than 10 pixels, or higher for strong nonlinear distortions). One marker, which has only four corners, always fits a homography perfectly, so this also implies that the minimum number of detected markers must be two. While an RMSE of 10 pixels is still unacceptable for calibration, the hope is that updating the prewarp with this rough estimate moves the markers to a better position in the next captured image.

Automatic Image Selection

Eventually, the machine should be able to detect a lot of the projector and printed markers (*e.g.*, over 50% for each). Also, if the corners of the calibration patterns have not moved much from the last captured image (*e.g.*, less than 5 pixels on average, depending on the slight movements from the user or oscillations when updating and applying the prewrap), then the system assumes that the patterns are aligned well enough, their motion blur negligible,

and saves the detected corners for later calibration. Nevertheless, if the user does not move the board, it would keep saving corners, but similar images are not useful for calibration. For this reason, we added another criterion. The system only selects an image if, since the last saved one, the corners of the calibration patterns moved a lot (*e.g.*, more than 50 pixels on average). When enough images have been selected (*e.g.*, 10 images), then all the saved corners go to the final stage: camera and projector calibration.

3.1.5 Calibration Algorithm

Once a sufficient amount of accurate corners have been accumulated, the calibration algorithm can process them directly. Using only the corners from printed markers $\mathbf{x}_\mathrm{b}$ and their detected corners $\mathbf{x}_\mathrm{c}$, OpenCV [8] can calibrate the camera with no difficulties using Zhang's method [39]. For the projector, on the other hand, we must first remove the linear (projective) and nonlinear (radial, etc.) distortions of the camera from the projector corners. Note that removing projective distortion is equivalent to applying the homography H_bc of Equation 3.5 to the imaged corners. Using such undistorted corner points $\mathbf{y}'_\mathrm{b}$ on the board and corresponding $\mathbf{y}'_\mathrm{p}$ on the image plane of the projector, we can apply the same method to calibrate the projector. Figure 3.6 illustrates the dataflow involved.

Although we have been writing about corners all along, we can also calibrate using the *center* of each marker. A center is defined as the physical point present in the middle, at the intersection of the diagonals. When nonlinear distortions are small, as is usually the case of projectors, Fiala and Shu [14]

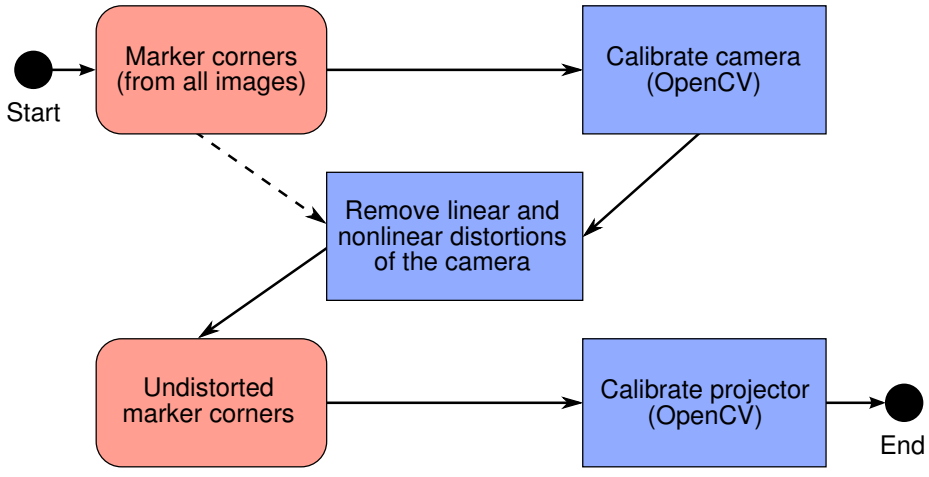


Figure 3.6: Dataflow diagram for calibrating the camera and the projector.

showed that we can obtain better results using centers than corners. This happens because corners that are not perfectly in focus are not accurately extracted. This is especially relevant in the case of projectors, since they usually have a narrow depth of field. To illustrate the problem, Figure 3.7 shows the lower left area of the detected markers in Figure 3.5: The edges of the defocused projector markers make them appear larger than they really are. Using what amounts to the average of the four corners lowers this bias.

3.2 Color Calibration

Once the geometric parameters of both the camera and projector have been estimated, the color calibration uses them to understand which portion of the calibration board can be covered by the projector. We then define that portion as a surface of white reference reflectance $\mathbf{p}_s = (1, 1, 1)$, information required to calibrate the colors. For simplicity, we reuse the same marked board (Figure 3.8). Further, the printed markers allow the machine to extract blank spots automatically, as shown in Figure 3.9. One may notice a vignetting effect, since the points in the middle are in reality closest to the projector. During calibration, if the board remains more or less at the same position, this reduction of brightness can be ignored as the algorithm only cares about the average.

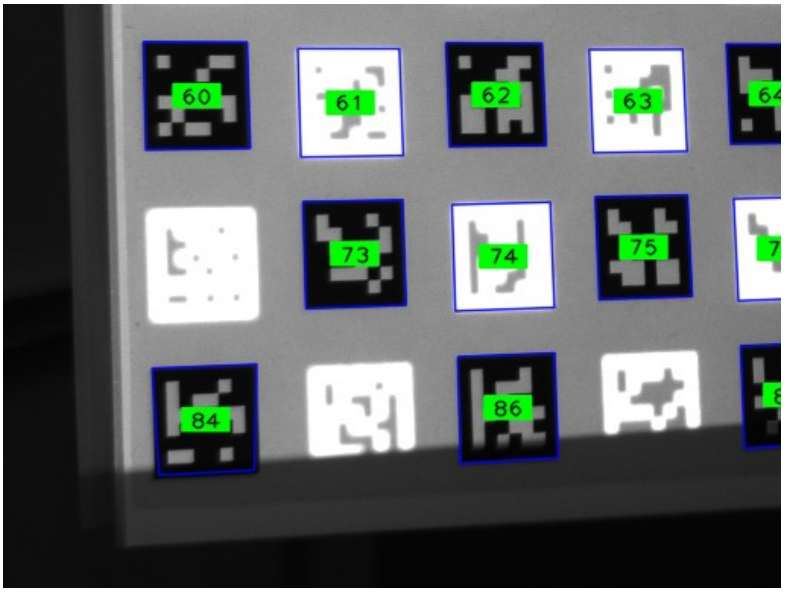


Figure 3.7: Lower left area of the detected markers from Figure 3.5. We can observe severe defocusing of projector markers.

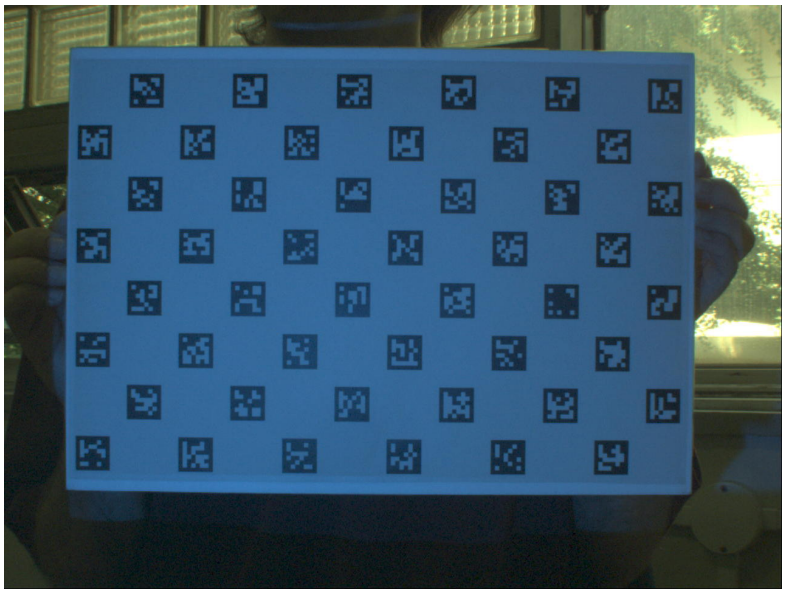


Figure 3.8: Sample camera image captured during color calibration when the projector displays the color blue at maximum intensity.

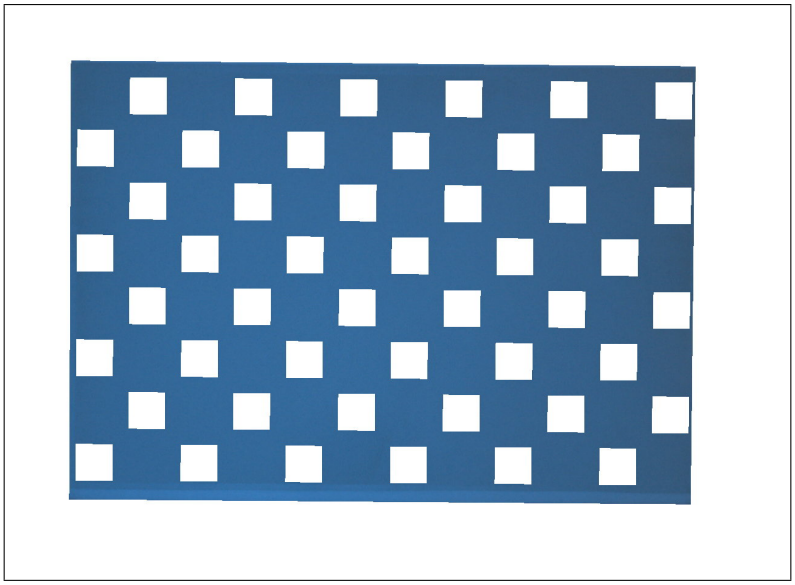


Figure 3.9: Blank area automatically extracted from Figure 3.8, whose average is used during color calibration.

The procedure itself goes as follows. First, the user must disable all onboard processing of the camera (no brightness, sharpness, white balance, hue, saturation, gamma, gain, etc.), and has to manually adjust the aperture and set the electronic shutter to a short enough exposure to avoid saturation at the highest projector intensity. Secondly, to find the camera noise bias $\mathbf{b}$ of Equation 2.4, one must blind the sensor and take the intensity values of the resulting image. We found it to be zero for all pixels of our camera, and in general we think it may safely be ignored as well. Next, by placing the calibration board in front of the projector and the camera, the system can detect its presence and start displaying and capturing an array of colors for calibration. For each color, it initially finds the pose of the calibration board via the detected markers, and assumes a gain $g = 1$. The pixel intensities are then averaged over the blank area of the board (Figure 3.9). Finally, the color mixing matrix X and the ambient light $\mathbf{a}$ remain the only two unknown parameters of Equation 2.4. They can be estimated by linearly regressing a sufficient number of samples. We found it enough to take a total of $4 \times 4 \times 4 = 64$ samples at constant intervals from the RGB color space of the projector as shown in Figure 3.10.

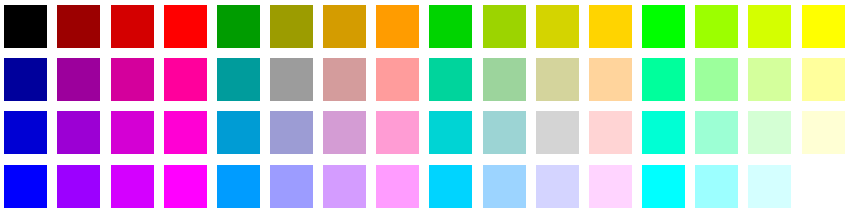


Figure 3.10: Samples from the RGB color space of the projector used during color calibration.

Chapter 4

Direct Image Alignment

Based on the simplifying assumptions mentioned above, the described models, and the parameters obtainable via calibration, we designed an iterative algorithm to directly align images of projector-camera systems. In this section, we describe the cost function and the algorithm to minimize it, which can hopefully converge to the correct alignment under normal circumstances. Following this, we provide details as to how one may initialize the algorithm for a new surface plane. However, we first assume that the parameters $\mathbf{n}$ and reflectance $\mathbf{p}_\mathrm{s}$ of the surface plane found during initialization are known.

4.1 Cost Function

We want to find an optimal set of parameters that best align the images. Intuitively, these parameters should minimally model the relative motion between the surface plane and the projector-camera system, since all other parameters are determined either during calibration or initialization. Looking back at Equations 2.6 and 2.8, these unknown parameters relating to motion are R_s and $\mathbf{t}_\mathrm{s}$, a typical egomotion problem. This represents six degrees of freedom. Four other degrees of freedom come from the gain g and the ambient light $\mathbf{a}$, which we assumed may change during motion, for a total of ten degrees of freedom and thus ten parameters to optimize. We can then formulate our goal into a cost function to minimize, which basically consists of the norm of the residual between the observed camera image and the expected image as defined by Equation 2.9, *i.e.*:

$$\min_{\mathrm{R}_\mathrm{s},\mathbf{t}_\mathrm{s},g,\mathbf{a}} \sum_{\mathbf{x}_\mathrm{c}} ||\mathbf{p}_\mathrm{c}(\mathbf{x}_\mathrm{c}) - \mathbf{p}_\mathrm{s}(\mathbf{w}_\mathrm{s}(\mathbf{x}_\mathrm{c}; \mathrm{R}_\mathrm{s}, \mathbf{t}_\mathrm{s})) \times [\ g\mathrm{X}\mathbf{p}_\mathrm{p}(\mathbf{w}_\mathrm{p}(\mathbf{x}_\mathrm{c}; \mathrm{R}_\mathrm{s}, \mathbf{t}_\mathrm{s})) + \mathbf{a}\ ] - \mathbf{b}||^2, \tag{4.1}$$

adding up to the maximum likelihood estimate (MLE) under the assumption of Gaussian noise. We chose the two-norm to simplify the implementation of the traditional Gauss-Newton optimization algorithm favored for image alignment problems [5], but more robust norms could also be used.

4.2 Minimization

To minimize the cost function of Equation 4.1, while one could optimize the 3D motion parameters directly, because of the inherent ambiguity between 3D rotation and translation, the convergence properties lack robustness [4]. From our own experiments, we also came to the same conclusion. For this reason, we instead decided to optimize the 2D homography $\mathrm{H_{sc}}$ of Equation 2.8 using a four-point parametrization [4], which does not exhibit this problem and from which we can easily extract the desired motion $\mathrm{R_s}$ and $\mathbf{t}_\mathrm{s}$ [36]. This however raises the number of parameters from six to eight, while the actual 3D motion remains at six degrees of freedom. To gain more robustness, we implemented constrained Gauss-Newton optimization using a Lagrange multiplier [26] to limit results to physically possible transformations. More precisely, we based the constraint on the Frobenius norm

$$|| \mathrm{H_{sc}} - \mathrm{K_c} \left(\mathrm{R_s} - \mathbf{t}_\mathrm{s}\mathbf{n}^\mathrm{T} \right) \mathrm{K_c^{-1}}||_F = 0. \tag{4.2}$$

While we have not done extensive testing and can only hypothesize about its effect on the convergence properties, we noted that on average it shaved about 10% off the number of iterations required to converge.

For simplicity and contrarily to Lucas and Kanade [5], we decided to evaluate the Jacobians numerically. This avoids the need to precompute image gradients and to analytically derive the cost function. Computationally, it does not appear to be less efficient. While the more usual algorithm would warp per iteration only four images instead of sixteen (two homographies), it would require twice the number of image multiplications to evaluate the derivatives and twice the amount of (cache) memory to hold image gradients. Nevertheless, we plan to investigate this in the future.

For additional robustness and performance, we implemented the traditional coarse-to-fine multiresolution estimation, where each level higher in the resolution pyramid is first smoothed with a 5×5 Gaussian filter and then subsampled by a factor of two. We found that five levels gave best results. For an initial image of 1024×768 pixels, this means low resolution images down to 64×48 pixels.

4.3 Initialization

The above minimization algorithm works only if the surface plane parameters $\mathbf{n}$ are known. The reflectance map $\mathbf{p}_\mathrm{s}$ also needs to be acquired somehow. Equation 2.4 shows that solving for the two unknowns $\mathbf{p}_\mathrm{s}$ and $\mathbf{a}$ requires capturing at least two images from the camera. Although there are undoubtedly many possible ways to perform initialization, we designed an approach that can cope with small movements, allowing users to hold the surface plane in their hands. This approach works in three phases, by first estimating the ambient light $\mathbf{a}$, then the reflectance map $\mathbf{p}_\mathrm{s}$ and finally the geometric plane parameters $\mathbf{n}$.

Concretely, the procedure goes as follows. It starts by the projection and the capture as fast as possible of three consecutive images, to obtain images with the smallest interframe motion possible, from which the user may select a region of interest. For the first image, the system sets the projector color $\mathbf{p}_\mathrm{p}$ to zero, and for the second, to maximum intensity $\mathbf{p}_\mathrm{p}^\mathrm{max}$. (More details about the third image below.) It can then estimate the ambient light using the first two images by defining the reference gain $g = 1$ and isolating $\mathbf{a}$ from Equation 2.4:

$$\mathbf{a} = \mathrm{X}\mathbf{p}_\mathrm{p}^\mathrm{max}\frac{\mathbf{p}_\mathrm{c}^1 - \mathbf{b}}{\mathbf{p}_\mathrm{c}^2 - \mathbf{p}_\mathrm{c}^1}, \tag{4.3}$$

where $\mathbf{p}_\mathrm{c}^1$ is the color from the first image, and $\mathbf{p}_\mathrm{c}^2$, from the second image. To reduce the undesirable effect of motion, we use at this phase severely smoothed versions of the first two images, for example by convolving with a 51×51 Gaussian kernel, which is reasonable since we assumed that ambient light varies only globally. This assumption also allows the system to ignore points where $\mathbf{p}_\mathrm{c}^2 - \mathbf{p}_\mathrm{c}^1$ is too close to zero and to evaluate only the average.

The second phase consists of using the second image, this time the original crisp version, along with the estimated ambient light to recover a crisp reflectance map

$$\mathbf{p}_\mathrm{s} = \frac{\mathbf{p}_\mathrm{c}^2 - \mathbf{b}}{\mathrm{X}\mathbf{p}_\mathrm{p}^\mathrm{max} + \mathbf{a}}. \tag{4.4}$$

For the third phase, the idea is to actually run the minimization algorithm described in the previous subsection, optimizing not only for g, H_sc and $\mathbf{a}$, but also for $\mathbf{n}$. As initial values, one can reasonably set g to 1, H_sc to I (the origin), reuse the ambient light computed in the first phase, and decide upon application-dependent plane parameters, for example frontoparallel at a distance of one meter. For the algorithm to converge properly though, some sort of texture needs to be displayed on the surface plane. We chose a fractal image, as shown in Figure 4.1, orthogonally aligned to the

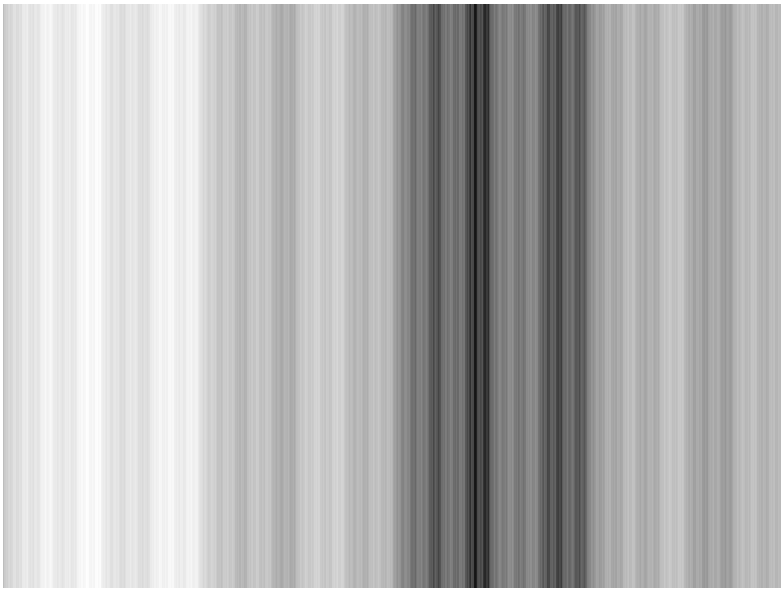


Figure 4.1: Fractal image displayed at the third phase of the initialization.

epipolar lines. Because this texture contains the same pattern at all scales, running the minimization algorithm at multiple resolutions should allow for easy and accurate convergence over a wide range of displacements. More formally, assuming epipolar lines aligned with the x-axis, for image coordinates $x \in [x_1, x_2]$ the intensities assigned to the projector image equal

$$\mathbf{p}_{\mathrm{p}}(x, y) = \begin{cases} f(x, x_1, x_m, 0, 0, \quad 1) & \text{if } \; x_1 \leq x < x_m \\ f(x, x_m, x_2, 0, 0, -1) & \text{if } \; x_m \leq x \leq x_2 \end{cases}, \tag{4.5}$$

whose amplitude is scaled appropriately, and where

$$f(x, x_1, x_2, y_1, y_2, n) = \begin{cases} f(x, x_1, x_m, y_1, y_m, \quad n') & \text{if } \; x_1 \leq x < x_m \\ y_m & \text{if } \; x = x_m \\ f(x, x_m, x_2, y_m, y_2, -n') & \text{if } \; x_m < x \leq x_2 \end{cases}, \tag{4.6}$$

$$x_m = \frac{x_1 + x_2}{2}, \quad y_m = \frac{y_1 + y_2}{2} + n, \quad n' = \frac{n}{\sqrt{2}}. \tag{4.7}$$

Although the barlike effect may give the impression of an image used for structured light, the approach is totally different as our pattern contains no code and tolerates well any surface texture.

Finally, the resulting plane parameters $\mathbf{n}$ might be slightly off from the values in the second image, from which the system actually derives the reflectance map $\mathbf{p}_\mathrm{s}$. We need to transform it using the inverse of R_s and $\mathbf{t}_\mathrm{s}$ to recover $\mathbf{n}_0$ of the second image. During image alignment in subsequent frames, $\mathbf{n}$ gets calculated by applying to $\mathbf{n}_0$ new camera motions R_s and $\mathbf{t}_\mathrm{s}$, which is the reason why the warping function $\mathbf{w}_\mathrm{p}(\mathbf{x}_\mathrm{c}; \mathrm{R}_\mathrm{s}, \mathbf{t}_\mathrm{s})$ depends on them.

Chapter 5

Results

The description of our methods is now complete, and we present here some results. We first programmed applications in Java that integrate ARToolKitPlus, OpenCV, PGR FlyCapture, and libdc1394 as appropriate to implement the procedures and algorithms described in this paper, then performed various tests running the software on a Dell Vostro 400 computer with an Intel Core 2 Quad Q6600 2.4 GHz CPU, and obtained the results described in the following sections for the geometric and color calibration as well as the image alignment algorithm.

5.1 Geometric Calibration

For geometric calibration, our test hardware consisted of a NEC LT157 (1024×768 color LCD) projector and a PGR Grasshopper GRAS-14S5M-C (1280×960 grayscale CCD) camera attached to a Fujinon HF16SA-1 (16 mm) lens. As calibration target, we used the calibration patterns of Figure 3.3, laser printed the first one on a B4 size sheet of paper and pasted it on a (mostly) flat foam board.

Table 5.1: Reprojection RMSE, root mean square errors, in pixels after calibrating using our method with either corners or centers.

	Reprojection RMSE (pixels)	
	Camera	Projector
Corners	0.43	0.66
Centers	0.33	0.20

Table 5.2: Reprojection RMSE, root mean square errors, in pixels of other methods. For tidiness, we cite only the first author.

Authors	Reprojection RMSE (pixels)	
	Camera	Projector
Ashdown [1]	0.25	0.47
Audet [2]	0.23	0.52
Drouin [12]	N/A	0.27
Gao [16]	0.63	0.43
Gao [17]	0.15	0.24
Griesser [18]	< 0.4	< 1.5
Kimura [22]	≈ 0.3	≈ 0.4
Legarda-Sáenz [23]	0.60	0.79
Li [25]	0.094	0.15
Zhang [38]	≈ 0.4	≈ 0.6
Average	0.34	0.54

Camera capture via software trigger took on average 158 ms, the display delay was about 64 ms, and the processing ran on only one core in approximately 350 ms, for a total of 572 ms per iteration. The test user could fully calibrate both camera and projector within about 30 seconds of manipulation. Figure 1.1 shows him in action on the casually installed system. The full sequence can be downloaded from
`http://www.ok.ctrl.titech.ac.jp/~saudet/procams2009.mp4` .

The reprojection errors, as listed in Table 5.1, also indicate that we were able to obtain an accurate subpixel calibration from this test session, which we could easily reproduce at will. Moreover, using a flatter wooden board, we could obtain better results of less than 0.15 pixels for both camera and projector. The errors also show that, as predicted, because the projector has a narrower depth of field than the camera, it benefits more from the use of marker centers.

For quick comparison, we summarize in Table 5.2 other reprojection errors reported in the literature. For publications with more than one set of numbers, we took only the best.

Table 5.3: Sample output of the color calibration, with bias $\mathbf{b}$ = zero.

Color mixing matrix X			Ambient light $\mathbf{a}$
0.3949	0.04552	0.008233	5.947
0.05504	0.8004	0.08908	9.551
0.0009641	0.02049	0.2771	3.114

5.2 Color Calibration

We could also obtain accurate results for color calibration. In this case, our test hardware consisted of a NEC LT157 (1024×768 color LCD) projector and a PGR Flea2 FL2G-13S2C-C (1280 × 960 Bayer color CCD) camera attached to a Cosmicar C814 (8 mm) lens, and we used the same calibration board as above. We placed in Table 5.3 a sample color mixing matrix obtained with our hardware. As shown in Figure 5.1, we typically obtain a root mean square error (RMSE) of approximately 0.5% of the total intensity range, and a correlation coefficient R^2 of almost one. This validates the assumptions of linear response of the camera sensor and of the sRGB response curve of the projector. One can observe that the green channel of the camera sensor is the most sensitive and the blue channel, the least, as expected from the specifications of the sensor.

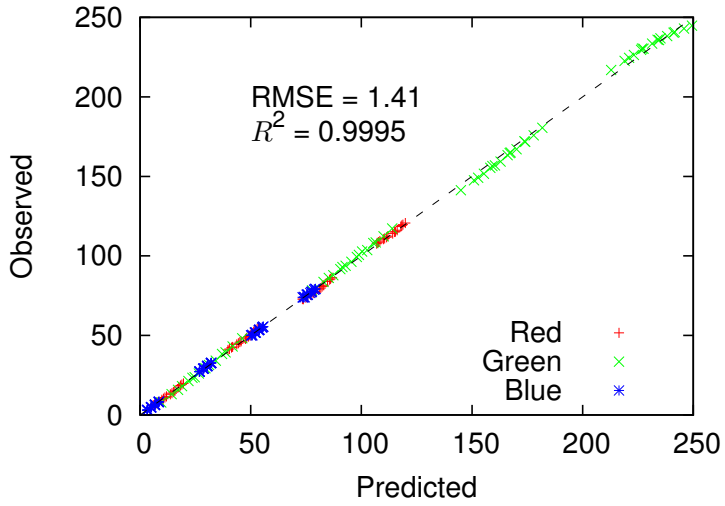


Figure 5.1: Predicted and observed intensities of the camera. Predicted values computed as per Equation 2.4 using parameters of Table 5.3. Observed values come from data taken during the color calibration using the sampling pattern discussed in Section 3.2.

5.3 Image Alignment

For image alignment, our test hardware consisted of a Casio XJ-S68 (1024×768 color DLP) projector, and a PGR Flea2 FL2G-13S2C-C (1280×960 Bayer color CCD) camera attached to a Pentax H1212B (12 mm) lens. For the surface planes, we inkjet printed the patterns described below on A4 size sheets of paper and pasted them on (mostly) flat foam boards. We conducted two sets of experiments, one to measure accuracy quantitatively by comparing alignment results with easy to detect markers and the other to demonstrate real-time operation and support for arbitrary textures.

As representative of the current methods, we chose ARToolKitPlus [37], plus OpenCV [8] for subpixel estimation, and compared its results with ours. We put the texture of Figure 5.2 on the planar surface. This fractal pattern is the 2D equivalent of Equation 4.5, but where triangles split the space at each recursion. Obviously, we also used Figure 4.1 for the projector pattern in an attempt to extract the best accuracy. On the resulting captured reflectance image, the markers delimited an initial region of interest (ROI) of 197255 pixels. On each frame, we left our algorithm run until ten iterations in a row did not lower the error of Equation 4.1 by more than 10%, taking on average a total of 2450 ms. Table 5.4 lists the time taken by each iteration at different levels of the resolution pyramid, where the warping function alone took most of the time. From this, Figure 5.4 shows the difference in pixels between our results and the centers of the four markers over the whole

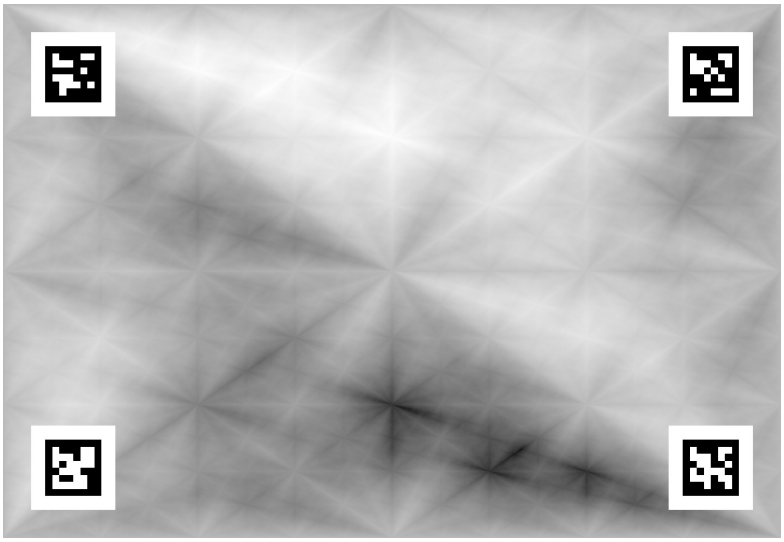
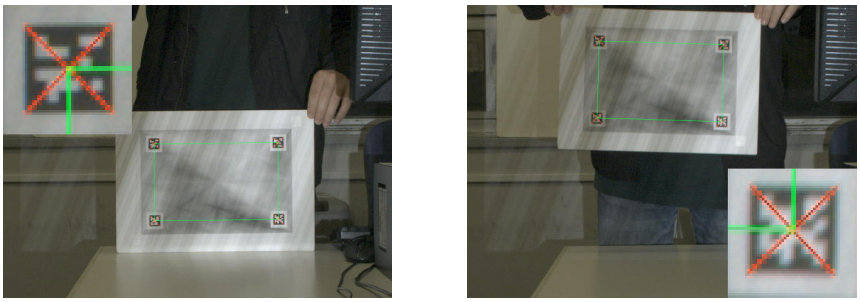
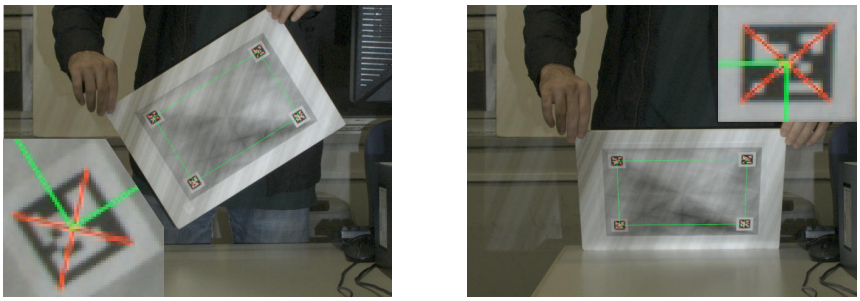


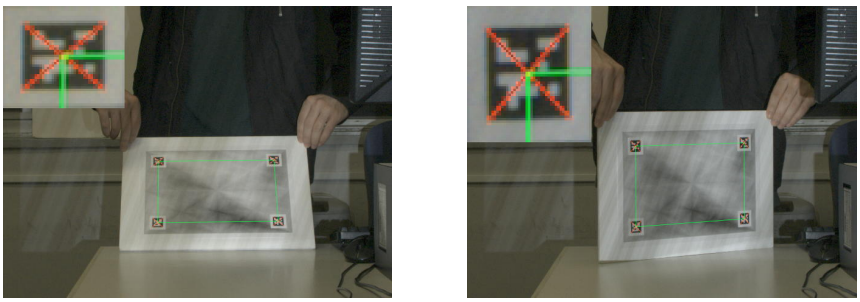
Figure 5.2: The printed triangular fractal pattern along with four markers that we used to obtain accuracy data.



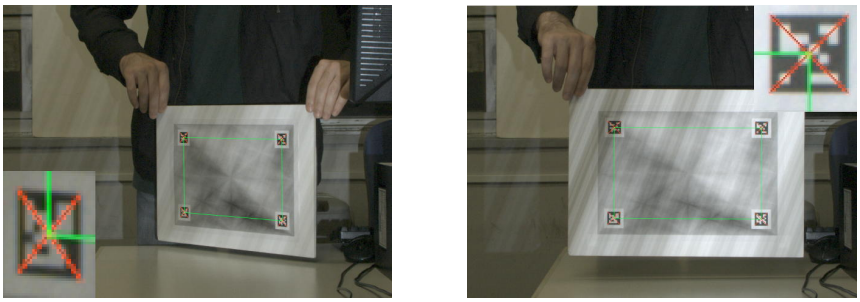
(a) frame 10 (b) frame 50



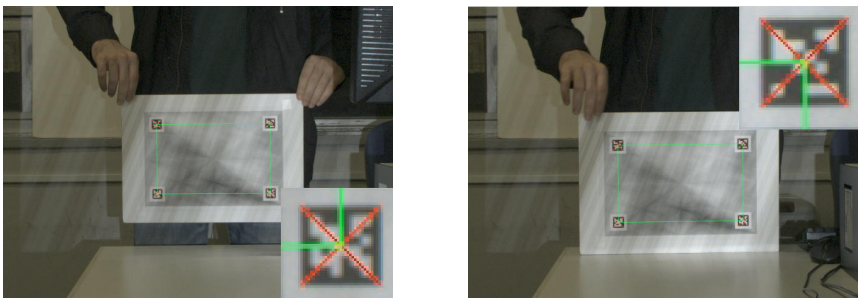
(c) frame 110 (d) frame 150



(e) frame 180 (f) frame 210



(g) frame 240 (h) frame 270



(i) frame 300 (j) frame 340

Figure 5.3: Frames from the marker test video with offline drawn red crosses representing the detected markers and green rectangles denoting the corresponding regions aligned by our program. The rectangle corners match the crosses with subpixel accuracy as shown in the blowups.

Table 5.4: Average time per iteration of the marker test.

Pyramid Level	Camera Resolution	Projector Resolution	Average Time (ms)
0	1280×960	1024×768	168
1	640×480	512×384	51
2	320×240	256×192	23
3	160×120	128×96	13
4	80×60	64×48	11

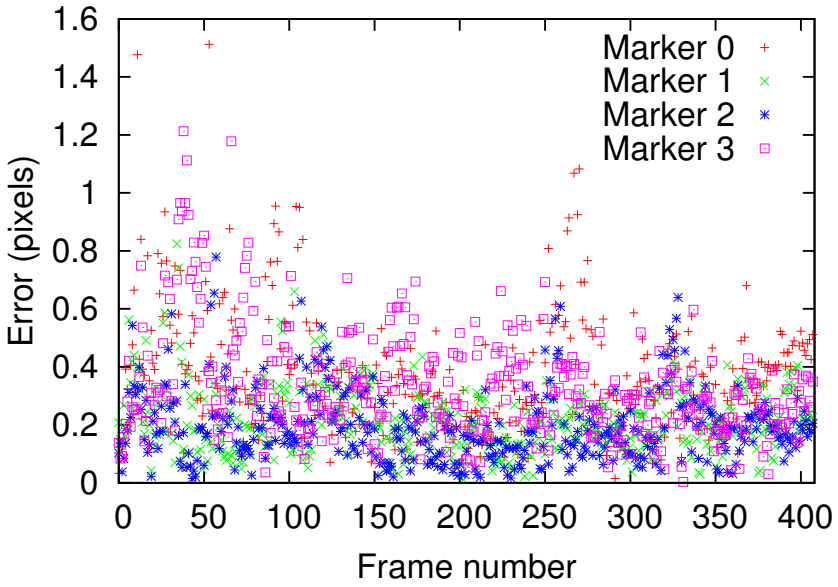
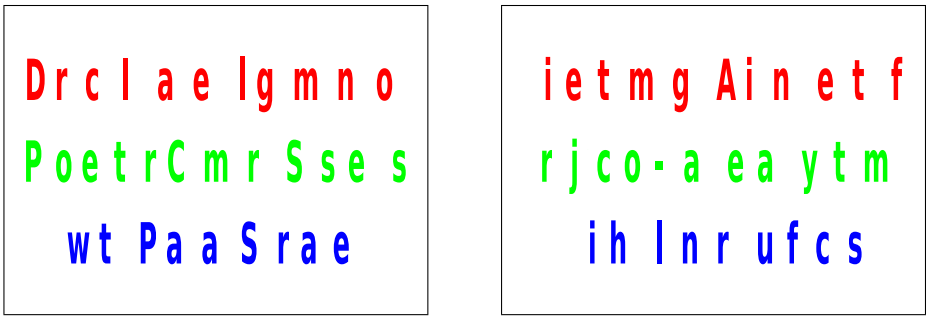


Figure 5.4: Difference in pixels between our results and the centers of the markers.

test video. Although markers do not provide the ground truth either, we consider the difference as errors of our algorithm since the error of corner estimation should be less than 0.10 pixels [8]. The total root mean square error (RMSE) sums up to 0.33 pixels, which even includes portions of motion blur (*e.g.*, Figure 5.3(b)) and images violating the assumed single gain g, especially true of slanted planes. We placed shots of the frames as captured by the camera in Figure 5.3. The full sequence can be downloaded from `http://www.ok.ctrl.titech.ac.jp/~saudet/markers_2009-11-14.mp4` .

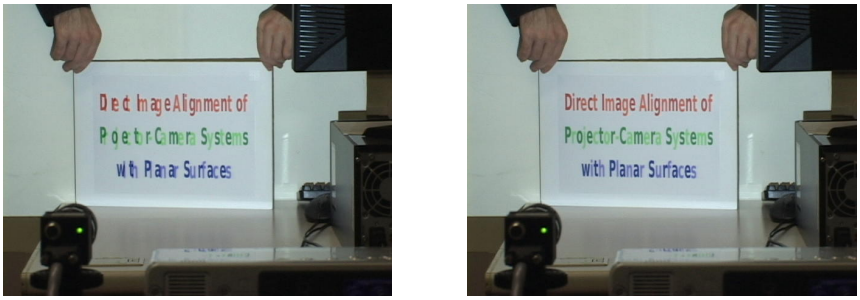
To show that our program can also work with arbitrary textures and almost in real time, we tested it using the images shown in Figure 5.5, the first pasted on the surface plane and the second displayed by the projector. More precisely, after each frame, the system warped the projector pattern to achieve a geometric correction on the physical plane by applying the homography $\mathrm{H_{ps}} = \mathrm{H_{pc}}\mathrm{H_{sc}^{-1}}$, which transforms points from the surface to the projector. (Although we could also correct the projector colors to match the printed one, we intentionally left them as is to differentiate them.) Figure 1.1 shows the system in action. We placed more shots of the demo video in Figure 5.6. The full sequence can be downloaded from `http://www.ok.ctrl.titech.ac.jp/~saudet/procamtracker_2009-11-14.mp4` . We limited to 300 ms the amount of time the program could spend iterating, which in reality ran on average for 374 ms. Camera capture via software trigger took a mean of 53 ms, the projector display delay was about 66 ms, and updating the projector and camera images took approximately 80 ms. This gives a total of 573 ms on average for each frame, close to two frames per second.

In both cases, the algorithm successfully converged given any displacements reasonable for a direct alignment method [5]. We also found that the single gain g used for the entire image was sufficient when all points of the surface plane are not far from each other relatively to their distance from the projector, which may be impractical for some applications.

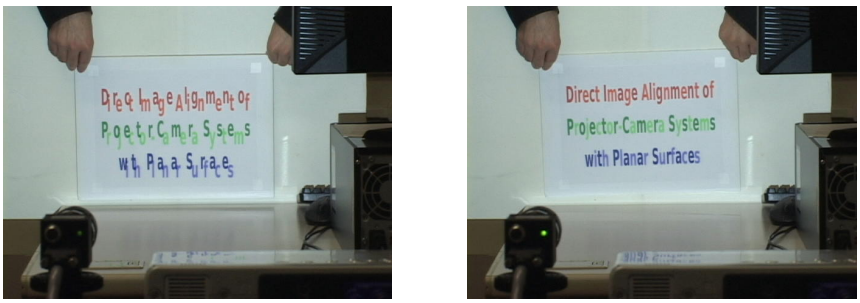


(a) Printed on the board. (b) Projector displayed.

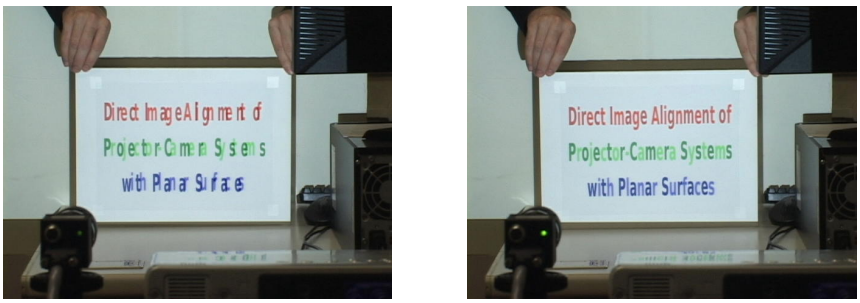
Figure 5.5: The images used for our demo video.



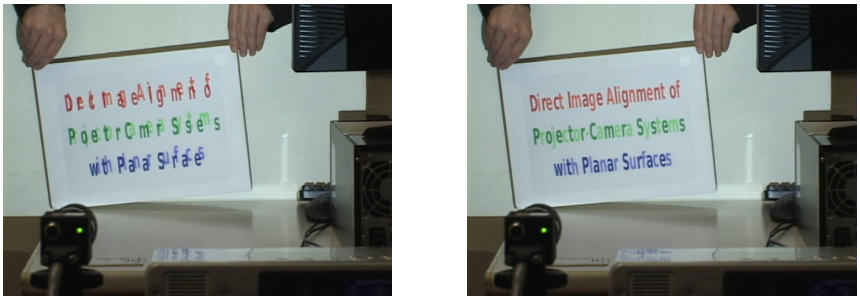
(a) frame 833 (b) frame 863



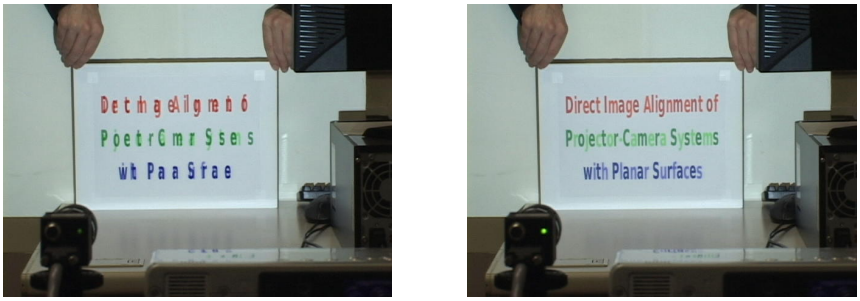
(c) frame 1057 (d) frame 1156



(e) frame 1523 (f) frame 1614



(g) frame 4142 (h) frame 4162



(i) frame 4596 (j) frame 4637

Figure 5.6: Frames from our demo video. We grouped them in pairs of a misaligned image caused by user motion followed by its correction.

Chapter 6

Discussion and Conclusion

Judging from our calibration results and comparing them to data available from previous methods, little of which include information about usability, we conclude that our method is the easiest to use. A calibration session that typically requires only a light board, a printed pattern, and less than one minute justifies in our eyes this designation. Furthermore, the subpixel accuracy and almost perfect color prediction achieved compare favorably with all previous methods. Further, since we based our geometric method solely on existing theory of camera calibration, assuming the noise characteristics of projectors are similar to cameras, published results for those [34, 39], with regards to the actual physical 3D accuracy, should translate similarly. However, this remains to be proven. For the current work, we fulfilled our main goal of user-friendliness. Moreover, we released the source code of our calibration application as free software and named it ProCamCalib, a screenshot of which we placed in Figure 6.1. It can be freely downloaded from `http://www.ok.ctrl.titech.ac.jp/~saudet/procamcalib/` .

As for the image alignment results, while they display good accuracy, we conclude that the computational complexity is the main limitation of our approach. To accelerate it, we plan on investigating mathematical approximations, algorithmic tricks, as well as hardware related optimizations. We have promising ideas on how to formulate an approximate model that the inverse compositional (IC) image alignment algorithm [5] could work with. To improve execution performance, we also tried to optimize the algorithm for graphics processing units (GPUs), but sadly failed in our first attempt at making it faster than the current CPU implementation. Fortunately, newer GPU architectures, such as Intel Larrabee and NVIDIA Fermi, feature multiple instruction and multiple data stream (MIMD) processing, which should allow easier and more efficient implementation of parallel tasks.

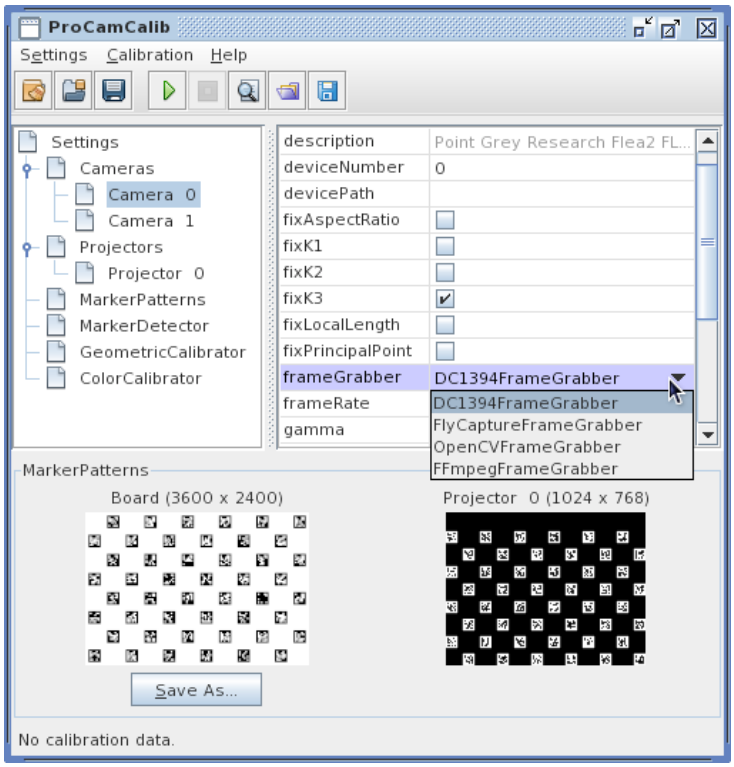


Figure 6.1: Screenshot of the main window of ProCamCalib, where all the settings can be adjusted. They are currently configured for calibrating two cameras and one projector.

The limited reflectance model represents another problem. To work around it, once the algorithm has been sufficiently accelerated, the surface may be divided in pieces each supporting a different gain and ambient light, as demonstrated by Silveira and Malis [31] for camera-only systems. Such an approach could even allow specularities.

Apart from these limitations, the algorithm can also be enhanced to support mostly planar 2.5D surfaces as well as piecewise planar objects by aligning multiple planes simultaneously, a widely adopted approach, as implemented by Sugimoto and Okutomi [33] for example. Further, a user may want to write new information on the surface. In that case, the system would need a way to gradually update the new reflectance properties.

Although much future work remains, we have demonstrated, at the very least, the feasibility of the approach. With some optimizations, we consider that, for future applications using projector-camera systems, direct alignment could become the basis of spatial augmented reality. Further, to encourage future research in this direction, we are making the whole software system available as open source on our Web site at
`http://www.ok.ctrl.titech.ac.jp/~saudet/procamtracker/` .

References

[1] Mark Ashdown and Yoichi Sato. Steerable Projector Calibration. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05) - Workshops (Procams 2005)*, volume 3, page 98. IEEE Computer Society, 2005.

[2] Samuel Audet and Jeremy R. Cooperstock. Shadow Removal in Front Projection Environments Using Object Tracking. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07) - Workshops (Procams 2007)*. IEEE Computer Society, June 2007.

[3] Samuel Audet and Masatoshi Okutomi. A User-Friendly Method to Geometrically Calibrate Projector-Camera Systems. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009) - Workshops (Procams 2009)*, pages 47–54. IEEE Computer Society, June 2009.

[4] Simon Baker, Ankur Datta, and Takeo Kanade. Parameterizing Homographies. Technical Report CMU-RI-TR-06-11, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2006.

[5] Simon Baker and Iain Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(1):221–255, March 2004.

[6] Deepak Bandyopadhyay, Ramesh Raskar, and Henry Fuchs. Dynamic Shader Lamps: Painting on Movable Objects. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR'01)*, page 207. IEEE Computer Society, 2001.

[7] Oliver Bimber and Ramesh Raskar. *Spatial Augmented Reality: Merging Real and Virtual Worlds*. A. K. Peters, Ltd., Natick, MA, USA, 2005.

[8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008.

[9] Dalit Caspi, Nahum Kiryati, and Joseph Shamir. Range Imaging With Adaptive Color Structured Light. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):470–480, 1998.

[10] Xinli Chen, Xubo Yang, Shuangjiu Xiao, and Meng Li. Color Mixing Property of a Projector-Camera System. In *Proceedings of the 5th ACM/IEEE International Workshop on Projector-Camera Systems (Procams 2008)*, pages 1–6. ACM, 2008.

[11] Paul E. Debevec and Jitendra Malik. Recovering High Dynamic Range Radiance Maps from Photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pages 369–378. ACM Press/Addison-Wesley Publishing Co., 1997.

[12] Marc-Antoine Drouin, Guy Godin, and Sébastien Roy. An Energy Formulation for Establishing the Correspondence Used in Projector Calibration. In *Proceedings of the Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, June 18–20, 2008.

[13] Mark Fiala. Comparing ARTag and ARToolkit Plus Fiducial Marker Systems. In *Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE 2005)*, page 6. IEEE Computer Society, October 1–2, 2005.

[14] Mark Fiala and Chang Shu. Self-identifying patterns for plane-based camera calibration. *Machine Vision and Applications*, 19(4):209–216, July 2008.

[15] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, Second edition, 1990.

[16] Wei Gao, Liang Wang, and Zhan-Yi Hu. Flexible Calibration of a Portable Structured Light System through Surface Plane. *Acta Automatica Sinica*, 34(11):1358–1362, November 2008.

[17] Wei Gao, Liang Wang, and Zhan-Yi Hu. Flexible Method for Structured Light System Calibration. *Optical Engineering*, 47(8):083602, August 2008.

[18] Andreas Griesser and Luc Van Gool. Automatic Interactive Calibration of Multi-Projector-Camera Systems. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '06), Procams 2006*, page 8. IEEE Computer Society, 2006.

[19] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Second edition, 2004.

[20] International Electrotechnical Commission. IEC 61966-2-1 (1999-10-18): Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB, 1999.

[21] Tyler Johnson and Henry Fuchs. Real-Time Projector Tracking on Complex Geometry Using Ordinary Imagery. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07) - Workshops (Procams 2007)*, pages 1–8. IEEE Computer Society, June 2007.

[22] Makoto Kimura, Masaaki Mochimaru, and Takeo Kanade. Projector Calibration using Arbitrary Planes and Calibrated Camera. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07) - Workshops (Procams 2007)*. IEEE Computer Society, June 2007.

[23] Ricardo Legarda-Sáenz, Thorsten Bothe, and Werner P. Jüptner. Accurate procedure for the calibration of a structured light system. *Optical Engineering*, 43(2):464, March 3, 2004.

[24] Bastian Leibe, Thad Starner, William Ribarsky, Zachary Wartell, David Krum, Brad Singletary, and Larry Hodges. The Perceptive Workbench: Towards Spontaneous and Natural Interaction in Semi-Immersive Virtual Environments. In *Proceedings of the 2000 IEEE Virtual Reality Conference*, pages 13–20. IEEE Computer Society, March 2000.

[25] Zhongwei Li, Yusheng Shi, Congjun Wang, and Yuanyuan Wang. Accurate calibration method for a structured light system. *Optical Engineering*, 47(5):053604, May 29, 2008.

[26] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, London, UK, 2003.

[27] Ramesh Raskar and Paul Beardsley. A Self-Correcting Projector. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, volume 2, pages 504–508. IEEE Computer Society, 2001.

[28] Ramesh Raskar, Jeroen van Baar, Paul Beardsley, Thomas Willwacher, Srinivas Rao, and Clifton Forlines. iLamps: Geometrically Aware and Self-Configuring Projectors. In *Proceedings of ACM SIGGRAPH 2003*, pages 809–818. ACM, 2003.

[29] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays. In *Proceedings of the 25th Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 179–188. ACM Press, 1998.

[30] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, February 2000.

[31] Geraldo Silveira and Ezio Malis. Real-time Visual Tracking under Arbitrary Illumination Changes. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*. IEEE Computer Society, June 2007.

[32] Peng Song and Tat-Jen Cham. A Theory for Photometric Self-Calibration of Multiple Overlapping Projectors and Cameras. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05) - Workshops (Procams 2005)*, volume 3, page 97. IEEE Computer Society, 2005.

[33] Shigeki Sugimoto and Masatoshi Okutomi. A Direct and Efficient Method for Piecewise-Planar Surface Reconstruction from Stereo Images. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*. IEEE Computer Society, June 2007.

[34] Wei Sun and Jeremy R. Cooperstock. An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques. *Machine Vision Application*, 17(1):51–67, 2006.

[35] Naoya Takao, Jianbo Shi, and Simon Baker. Tele-Graffiti: A Camera-Projector Based Remote Sketching System with Hand-Based User Interface and Automatic Session Summarization. *International Journal of Computer Vision*, 53(2):115–133, July 2003.

[36] Bill Triggs. Autocalibration from Planar Scenes. In *Proceedings of the 5th European Conference on Computer Vision (ECCV '98)*, volume I, pages 89–105. Springer-Verlag, 1998.

[37] Daniel Wagner and Dieter Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices. In *Proceedings of the 12th Computer Vision Winter Workshop 2007 (CVWW'07)*. Graz University of Technology, St. Lambrecht, Austria, February 6–8, 2007.

[38] Song Zhang and Peisen S. Huang. Novel method for structured light system calibration. *Optical Engineering*, 45(8):083601, August 2006.

[39] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.