Direct Image Alignment of Projector-Camera Systems with Planar Surfaces

Samuel Audet, Masatoshi Okutomi, and Masayuki Tanaka
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo, Japan
saudet@ok.ctrl.titech.ac.jp, mxo@ctrl.titech.ac.jp, and mtanaka@ctrl.titech.ac.jp

Abstract

Projector-camera systems use computer vision to analyze their surroundings and display feedback directly onto real world objects, as embodied by spatial augmented reality. To be effective, the display must remain aligned even when the target object moves, but the added illumination causes problems for traditional algorithms. Current solutions consider the displayed content as interference and largely depend on channels orthogonal to visible light. They cannot directly align projector images with real world surfaces, even though this may be the actual goal. We propose instead to model the light emitted by projectors and reflected into cameras, and to consider the displayed content as additional information useful for direct alignment. We implemented in software an algorithm that successfully executes on planar surfaces of diffuse reflectance properties at almost two frames per second with subpixel accuracy. Although slow, our work proves the viability of the concept, paving the way for future optimization and generalization.

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 [6], we instead use video projectors to display computer graphics or data onto surfaces, as exemplified in Figure 1. In this case, the appearance of the objects may be severely affected, requiring new methods.

Figure 1. Snapshot of our demo video showing the patterns of Figure 8, where the system has aligned the projector displayed pattern with the printed one.

1.1. Previous Work

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 [16], or a special tracking system not based on visible light, such as Dynamic Shader Lamps [4], 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 [17]. 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 [14] 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 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 [20] features a paper tracking algorithm that detects the orientation of a piece of paper or a clipboard inside images captured from 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. The Direct Approach

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. 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 explain in more details the design, including the system model and its simplifying assumptions, and then the 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.

2. System Model

A projector-camera system can be treated as a stereo pair, and multiple view geometry [11] 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 explaining 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 [12] 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 uniform diffuse reflectance, where the radiance changes smoothly 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 model geometrically both camera and projector devices, we use the familiar pinhole camera model [11]. Although originally designed for cameras, it also applies to projectors. It maps a 3D point $\mathbf{x}_3$ of the surface to

$$\mathbf{x}_2 = \mathrm{K}_{3\times3} \; (\; \mathrm{R}_{3\times3}\mathbf{x}_3 + \mathbf{t}_3 \;), \tag{1}$$

a 2D point on the image plane of the device, where K, the camera matrix, contains the internal projective 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

$$\begin{aligned} \mathbf{x}_\mathrm{c} &= \mathrm{K}_\mathrm{c} \; (\; \mathrm{I} \;\; \mathbf{x}_3 + \mathbf{0} \;\;), & (2) \\ \mathbf{x}_\mathrm{p} &= \mathrm{K}_\mathrm{p} \; (\; \mathrm{R}_\mathrm{p}\mathbf{x}_3 + \mathbf{t}_\mathrm{p} \;). & (3) \end{aligned}$$

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

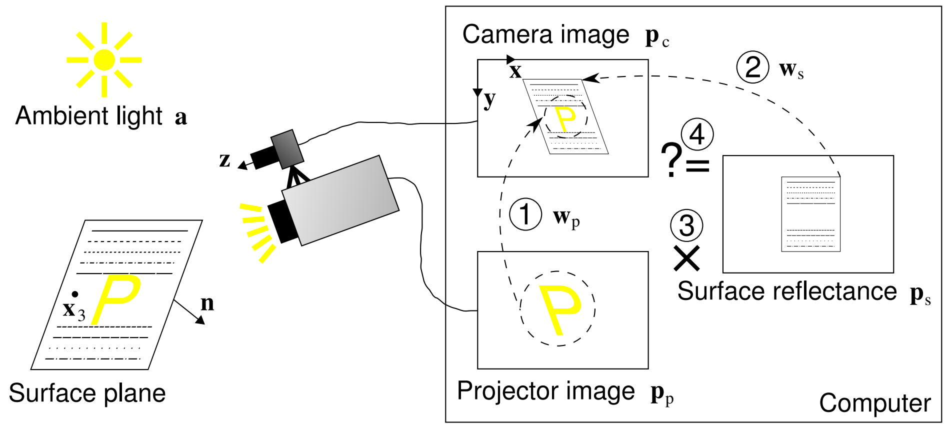


Figure 2. Sketch of the image formation model.

for simplicity to model only the relationship between devices. Other methods [8, 9, 10, 13, 19] require either control over the shutter speed of the camera, more than one projector, or an elaborate 3D contraption, but we wanted a model that we could calibrate without camera control and with only one projector and a planar surface, as follows.

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

$$\mathbf{p}_\mathrm{c} = \mathbf{p}_\mathrm{s} \times [\, g\mathrm{X}_{3\times 3}\mathbf{p}_\mathrm{p} + \mathbf{a}\,] + \mathbf{b}, \tag{4}$$

where g is the *gain* of the projector light, which we assume varies smoothly and uniformly with the distance and angle to the surface; X is the *color mixing matrix* as defined by Chen *et al*. [9], also called the "projector-camera coupling matrix" by Caspi *et al*. [8]; $\mathbf{a}$, 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 [13] 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.

2.4. 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 $\mathbf{t}_\mathrm{p}$) [1, 23] and color parameters (X and $\mathbf{b}$) [8, 9], but we assume linearized color responses from the start.

2.5. 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. 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 $\mathbf{n}^\mathrm{T}\mathbf{x}_3 + 1 = 0$, it follows that the 2D point $\mathbf{x}_\mathrm{p}$ from Equation 3 can also be expressed as

$$\begin{aligned} \mathbf{x}_\mathrm{p} &= \mathrm{K}_\mathrm{p}\,(\,\mathrm{R}_\mathrm{p}\mathbf{x}_3 + \mathbf{t}_\mathrm{p}\,) \\ &= \mathrm{K}_\mathrm{p}\,(\,\mathrm{R}_\mathrm{p}\mathbf{x}_3 - \mathbf{t}_\mathrm{p}\mathbf{n}^\mathrm{T}\mathbf{x}_3\,) \text{ since } \mathbf{n}^\mathrm{T}\mathbf{x}_3 = -1 \\ &= \mathrm{K}_\mathrm{p}\,(\,\mathrm{R}_\mathrm{p} - \mathbf{t}_\mathrm{p}\mathbf{n}^\mathrm{T}\,)\,\mathbf{x}_3 \\ &= \mathrm{K}_\mathrm{p}\,(\,\mathrm{R}_\mathrm{p} - \mathbf{t}_\mathrm{p}\mathbf{n}^\mathrm{T}\,)\,\mathrm{K}_\mathrm{c}^{-1}\,\mathbf{x}_\mathrm{c} \\ &= \mathrm{H}_\mathrm{pc}\,\mathbf{x}_\mathrm{c}, \end{aligned} \tag{5}$$

where the before last substitution comes from the fact that homogeneous vector $\mathbf{x}_\mathrm{c}$ from Equation 2 can be arbitrarily scaled to fit in. This shows that a camera pixel $\mathbf{x}_\mathrm{c}$ is transformed by a homography H_pc into the projector pixel $\mathbf{x}_\mathrm{p}$. From this we define the *warping function* from the camera image plane to the one of the projector:

$$\mathbf{w}_\mathrm{p}(\mathbf{x}_\mathrm{c}) \equiv \mathrm{H}_\mathrm{pc}\,\mathbf{x}_\mathrm{c}. \tag{6}$$

This corresponds to step 1 in Figure 2.

Similarly, one can map a camera image point $\mathbf{x}_\mathrm{c}$ onto a point $\mathbf{x}_\mathrm{s}$ 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, $\mathbf{t}_\mathrm{s}$ 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:

$$\begin{aligned} \mathbf{x}_\mathrm{s} &= \mathrm{K}_\mathrm{c}\,(\,\mathrm{R}_\mathrm{s}\mathbf{x}_3 + \mathbf{t}_\mathrm{s}\,) \\ &= \mathrm{K}_\mathrm{c}\,(\,\mathrm{R}_\mathrm{s}\mathbf{x}_3 - \mathbf{t}_\mathrm{s}\mathbf{n}^\mathrm{T}\mathbf{x}_3\,) \text{ since } \mathbf{n}^\mathrm{T}\mathbf{x}_3 = -1 \\ &= \mathrm{K}_\mathrm{c}\,(\,\mathrm{R}_\mathrm{s} - \mathbf{t}_\mathrm{s}\mathbf{n}^\mathrm{T}\,)\,\mathbf{x}_3 \\ &= \mathrm{K}_\mathrm{c}\,(\,\mathrm{R}_\mathrm{s} - \mathbf{t}_\mathrm{s}\mathbf{n}^\mathrm{T}\,)\,\mathrm{K}_\mathrm{c}^{-1}\,\mathbf{x}_\mathrm{c} \\ &= \mathrm{H}_\mathrm{sc}\,\mathbf{x}_\mathrm{c}, \end{aligned} \tag{7}$$

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:

$$\mathbf{w}_s(\mathbf{x}_c) \equiv \mathrm{H}_{sc}\, \mathbf{x}_c. \tag{8}$$

This corresponds to step 2 in Figure 2.

Finally, plugging these coordinates into the color model of Equation 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

$$\mathbf{p}_c(\mathbf{x}_c) = \mathbf{p}_s(\mathbf{w}_s(\mathbf{x}_c)) \times [\, g\mathrm{X}\mathbf{p}_p(\mathbf{w}_p(\mathbf{x}_c)) + \mathbf{a}\,] + \mathbf{b}. \tag{9}$$

This corresponds to the final third and fourth steps in Figure 2.

3. Alignment Algorithm

Based on the mentioned simplifying assumptions, the models illustrated above, and the parameters obtainable via calibration, we designed an iterative method to directly align images of projector-camera systems. In this section, we explain 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 $\mathbf{p}_s$ of the surface plane found during initialization are known.

3.1. Cost Function

To align best the images, we want to find an optimal set of parameters. 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 6 and 8, these unknown parameters relating to motion are R_s and $\mathbf{t}_s$, a typical egomotion problem, which represents six degrees of freedom. Four others 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 consequently ten parameters to optimize. We can thus formulate our goal into a cost function to minimize, which basically consists of the norm of the residual between the observed camera image $\bar{\mathbf{p}}_c$ and the expected image as defined by Equation 9, *i.e.*:

$$\begin{aligned}\min_{\mathrm{R}_s,\mathbf{t}_s,g,\mathbf{a}} \sum_{\mathbf{x}_c} ||\bar{\mathbf{p}}_c(\mathbf{x}_c) - \mathbf{p}_s(\mathbf{w}_s(\mathbf{x}_c;\mathrm{R}_s,\mathbf{t}_s))\times \\ [\, g\mathrm{X}\mathbf{p}_p(\mathbf{w}_p(\mathbf{x}_c;\mathrm{R}_s,\mathbf{t}_s)) + \mathbf{a}\,] - \mathbf{b}||^2,\end{aligned} \tag{10}$$

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 [3], but more robust norms could also be used.

3.2. Minimization

To minimize the cost function of Equation 10, while one could optimize the 3D motion parameters directly, because of the inherent ambiguity between 3D rotation and translation, the convergence properties lack robustness [2]. From our own experiments, we also came to the same conclusion. For this reason, we instead decided to optimize the 2D homography H_{sc} of Equation 8 using a four-point parametrization [2], which does not exhibit this problem and from which we can easily extract the desired motion R_s and $\mathbf{t}_s$ [21]. 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 [15] to limit results to physically possible transformations. More precisely, we based the constraint on the Frobenius norm

$$||\, \mathrm{H}_{sc} - \mathrm{K}_c\, (\, \mathrm{R}_s - \mathbf{t}_s\mathbf{n}^{\mathrm{T}}\,)\, \mathrm{K}_c^{-1}||_F = 0. \tag{11}$$

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 [3], 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.

3.3. Initialization

The above minimization algorithm works only if the surface plane parameters $\mathbf{n}$ are known. The reflectance map $\mathbf{p}_s$ also needs to be acquired somehow. Equation 4 shows that solving for the two unknowns $\mathbf{p}_s$ 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 shots, to obtain images with the smallest inter-frame 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.) Taking the first two images, it can then estimate the ambient light by defining the reference gain $g = 1$ and isolating $\mathbf{a}$ from Equation 4:

$$\mathbf{a} = \mathrm{X}\mathbf{p}_\mathrm{p}^\mathrm{max} \frac{\mathbf{p}_\mathrm{c}^1 - \mathbf{b}}{\mathbf{p}_\mathrm{c}^2 - \mathbf{p}_\mathrm{c}^1}, \tag{12}$$

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

$$\mathbf{p}_\mathrm{s} = \frac{\mathbf{p}_\mathrm{c}^2 - \mathbf{b}}{\mathrm{X}\mathbf{p}_\mathrm{p}^\mathrm{max} + \mathbf{a}}. \tag{13}$$

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 3, orthogonally aligned to the 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, the intensities assigned to the projector image for coordinates $x \in [x_1, x_2]$ equal

$$\mathbf{p}_\mathrm{p}(x, y) = \begin{cases} f(x, x_1, x_m, 0, 0, \quad 1) & \text{if } x_1 \le x < x_m \\ f(x, x_m, x_2, 0, 0, -1) & \text{if } x_m \le x \le x_2 \end{cases}, \tag{14}$$

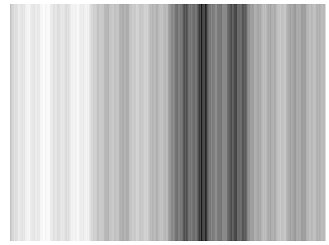


Figure 3. Fractal image displayed at the third phase of the initialization.

whose amplitude is scaled appropriately, and where

$$f(x, x_1, x_2, y_1, y_2, n) = \begin{cases} f(x, x_1, x_m, y_1, y_m, \quad n') & \text{if } x_1 \le x < x_m \\ y_m & \text{if } x = x_m \\ f(x, x_m, x_2, y_m, y_2, -n') & \text{if } x_m < x \le x_2 \end{cases}, \tag{15}$$

$$x_m = \frac{x_1 + x_2}{2}, \quad y_m = \frac{y_1 + y_2}{2} + n, \quad n' = \frac{n}{\sqrt{2}}. \tag{16}$$

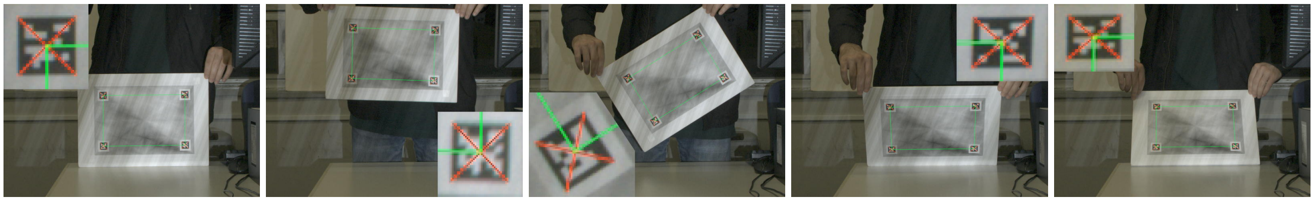
Although the barlike effect may give the impression of an image used for structured light, the approach is totally different as our pattern contains no code and tolerates well any surface texture.

Finally, because of user motion 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 alignment in subsequent frames, $\mathbf{n} = \frac{\mathrm{R}_\mathrm{s}\mathbf{n}_0}{1 - \mathbf{t}_\mathrm{s}^\mathrm{T}\mathbf{n}_0}$.

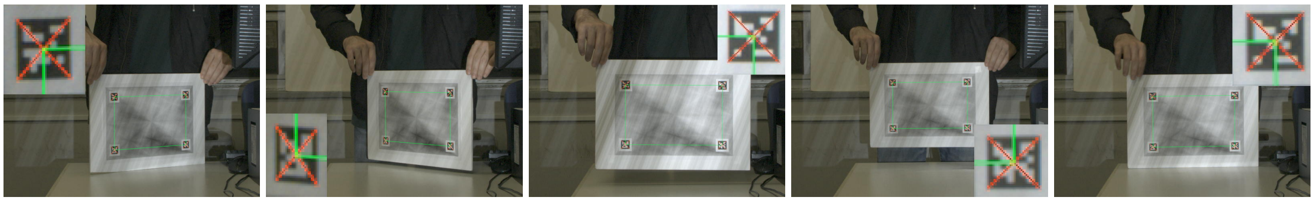
4. Results

The description of our method is now complete, and we present here some results. We programmed an application in Java, which integrates OpenCV and libdc1394 as appropriate, and that implements the procedures and algorithms introduced in this paper. 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, both connected to a Dell Vostro 400 computer with an Intel Core 2 Quad Q6600 2.4 GHz CPU. As surface plane, we inkjet printed patterns 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 [22], plus OpenCV [7] for subpixel estimation, and compared its results with ours. We put the texture of Figure 5 on the planar surface. This fractal pattern



(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 4. Frames from the markers test video with offline drawn red crosses representing 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.

is the 2D equivalent of Equation 14, but where triangles split the space at each recursion. Obviously, we also used Figure 3 for the projector pattern in an attempt to extract the best accuracy. On the resulting initialized 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 10 by more than 10%, taking on average

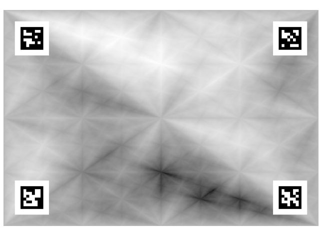


Figure 5. The printed triangular fractal pattern along with four markers that we used to obtain accuracy data.

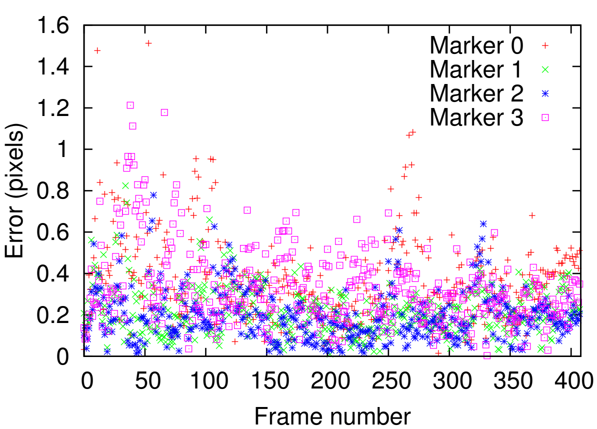


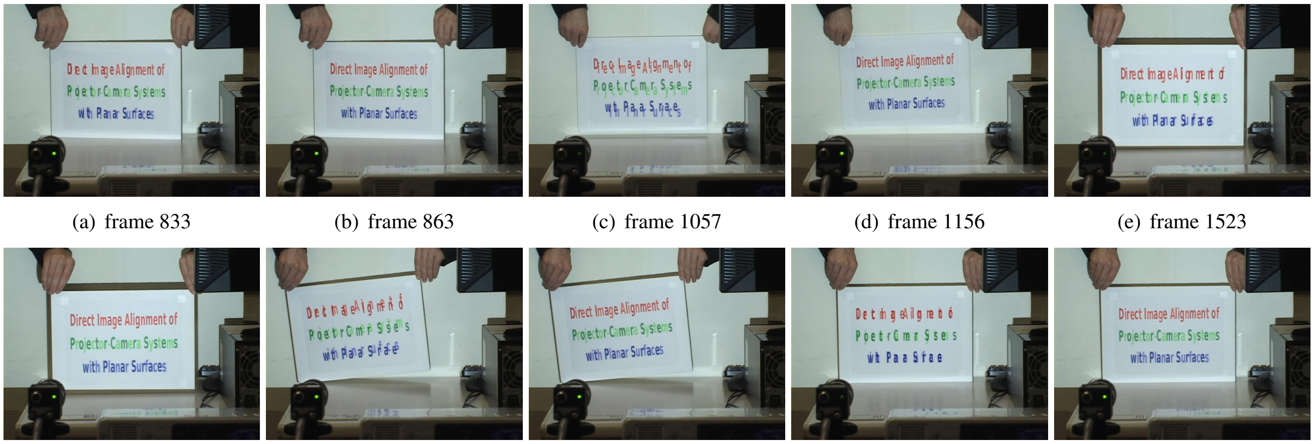
Figure 6. Difference in pixels between our results and the detected centers of the markers.

a total of 2450 ms. Table 1 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 6 shows the difference in pixels between our results and the centers of the four markers over the whole 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 [7]. The total root mean square error (RMSE) sums up to 0.33 pixels, which even includes portions of motion blur (*e.g.*, Figure 4(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 4. The full sequence can be found in the supplementary material as well as on our Web site: http://www.ok.ctrl.titech.ac.jp/~saudet/markerstest_2009-11-14.mp4 .

To show that our program can also work almost in real time even with poor and arbitrary textures that easily overlap on the surface, we ran it on the images shown in Figure 8, 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 $\mathrm{H}_{\mathrm{ps}} = \mathrm{H}_{\mathrm{pc}}\mathrm{H}_{\mathrm{sc}}^{-1}$, which transforms points from the surface to the projector. (Although we could also

Table 1. Average time per iteration of the markers 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



(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

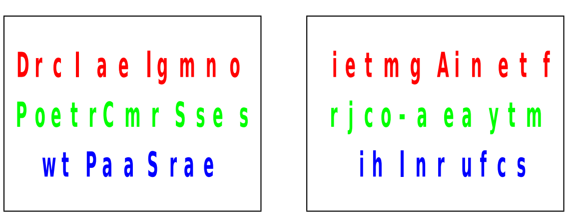
Figure 7. Frames from our demo video. We grouped them in pairs of a misaligned image caused by user motion followed by its correction.

correct the projector colors to match the printed one, we intentionally left them as is to differentiate them.) Figure 1 shows the system in action. We placed more shots of the demo video in Figure 7. The full sequence can be found in the supplementary material as well as on our Web site: 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 [3]. 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, otherwise vignettinglike effects obviously occur, a possibly impractical limitation for some applications.

5. Discussion and Conclusion

From the results, we conclude that the computational complexity is the main limitation of our approach. To



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

Figure 8. The images used for our demo video.

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 [3, 5] could work with. To improve execution performance, we also tried to implement 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 parallel tasks to run faster more easily.

The limited reflectance model presents another problem. To work around it, once the algorithm has been sufficiently accelerated, the surface may be divided in pieces each supporting different gain and ambient light, as demonstrated by Silveira and Malis [18] for camera-only systems. Such an approach could even allow specularities.

Apart from these limitations, the algorithm can also be enhanced to support more complex 3D surfaces, for example by using a piecewise planar model and aligning multiple planes simultaneously. 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 image 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:
http://www.ok.ctrl.titech.ac.jp/~saudet/procamtracker/ .

Acknowledgments

This work was supported by a scholarship from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of the Japanese Government.

References

[1] S. Audet and M. 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.

[2] S. Baker, A. Datta, and T. Kanade. Parameterizing Homographies. Technical Report CMU-RI-TR-06-11, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2006.

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

[4] D. Bandyopadhyay, R. Raskar, and H. 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.

[5] A. Bartoli. Groupwise Geometric and Photometric Direct Image Registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2098–2108, December 2008.

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

[7] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008.

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

[9] X. Chen, X. Yang, S. Xiao, and M. 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.

[10] P. E. Debevec and J. 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.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Second edition, 2004.

[12] 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.

[13] T. Johnson and H. 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.

[14] B. Leibe, T. Starner, W. Ribarsky, Z. Wartell, D. Krum, B. Singletary, and L. 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.

[15] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, London, UK, 2003.

[16] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. iLamps: Geometrically Aware and Self-Configuring Projectors. In *Proceedings of ACM SIGGRAPH 2003*, pages 809–818. ACM, 2003.

[17] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. 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.

[18] G. Silveira and E. 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.

[19] P. Song and T.-J. 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.

[20] N. Takao, J. Shi, and S. 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.

[21] B. 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.

[22] D. Wagner and D. 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.

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