Autocorrelation-based Fiducial Markers for Traceability

Ismail Bencheikh, Max Dunitz, Marie d'Autume, Enric Meinhardt-Llopis, Marc Pic, Gabriele Facciolo, Pablo Musé

Autocorrelation-based Fiducial Markers for Traceability

Ismail Bencheikh[1,3,*] Max Dunitz[1,3,*] Marie d'Autume[1] Enric Meinhardt-Llopis[1]
Marc Pic[3] Gabriele Facciolo[1,2] Pablo Musé[1,4]
[1]Université Paris-Saclay, ENS Paris-Saclay, CNRS, Centre Borelli, France
[2]Institut Universitaire de France (IUF) [3]Advanced Track & Trace, France
[*] Equal contributions. [4]IIE, Facultad de Ingeniería, Universidad de la República, Uruguay

Abstract

Classical approaches to the rectification of a single image of a product, without stereo correspondences, require spatial landmarks. These landmarks, constructed from high-contrast elementary shapes that can be detected with simple algorithms, are highly conspicuous. To rectify complex deformations, one can use chessboard patterns of markers with elements that break quadrilateral symmetry, such as the three eyes of a QR code. However, these marker boards are even more conspicuous than a single marker. In traceability applications, only one site of marking is used, limiting the complexity of the surface on which it can be read, and exposing the mark to deidentification attacks for diversion of the product to a grey market. We introduce a method for constructing stealth and robust fiducial markers that can be displayed across a surface, limiting exposure to marker tampering for product deidentification. These markers, which we refer to as self-rectifying textures, can be used to rectify complex deformations by solving an inverse problem rather than relying on pixel correspondences of conspicuous landmarks. These stealth textures place fiducial markers in the autocorrelation of the image. In this way, crops of the deformed texture can be rectified using only these spatially invariant statistical properties. Affine transformations of an image correspond to linear transformations of the autocorrelation, without phase component. Exploiting this fact, self-rectifying textures enable the local estimation of the differential of a planar deformation by identifying landmarks in the autocorrelation image, such as peaks, whose locations in the fronto-parallel view of the texture are known. The translation component can be recovered independently via phase correlation. A rectifying map, modulo translations, can also be fit directly to local observations of the differential of the deformation, without access to the rectified texture or need for phase correlation. Self-rectifying textures can be used for communication, watermarking, authentication, surface identification, calibration, and geometry processing.

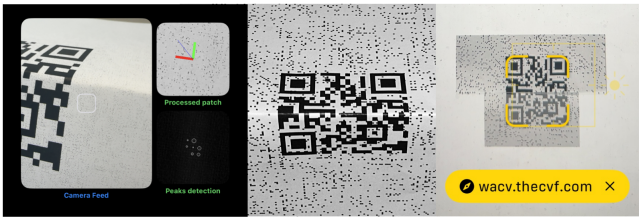


Figure 1. A self-rectifying texture and a QR code are printed on a piece of paper folded across a table edge. This deformation prevents the QR code from being read. By contrast, the real-time self-rectifying texture reader (left) can acquire the deformation differential (left lower inset) from the small patch being processed (left upper inset); from this single patch, after affine rectification, the traceability information is easily decoded. A photograph of the QR code spread across two surface faces (center) cannot be read. However, the geometric information exposed by the self-rectifying texture easily allows rectification of a homography for each planar surface, allowing the QR code to be read.

1. Introduction

Trademark holders, inventory managers, consumer advocates, safety and environmental regulators, customs officials, central banks, and tax authorities seek better technologies and standards to ensure product traceability and supply chain transparency [14, 18, 37]. As physical goods come in many forms, markings containing product information must be easily readable, even on complex surfaces. Traceability markers are common sites of tampering as product deidentification is often necessary for diversion to a grey market. As a defense against tampering, robust traceability marking requires redundancy–and thus, due to aesthetic constraints, stealth.

Fiducial markers (see Fig. 2) generally consist of high-contrast images of large, elementary objects like squares, enabling detection by simple algorithms [5, 17, 21, 43, 46].

They reveal pose information via pixel locations of landmarks–often the corners of a square marker's bounding box–along with elements that disrupt symmetries to distinguish the landmarks (such as the three "eyes" of QR code). Only the repetition of markers across a surface can endow landmark-based rectification methods with robustness to intense, localized occlusion due to deidentification. Such repetition often grants the ability to rectify planar deformations more complex than homographies. But repeated markers generally assume the form of chessboard-like arrays of high-contrast markers, known as marker boards [19]. We seek a configurable family of *inconspicuous* fiducial markers that, like marker boards, are geometrically informative and robust to deidentification occlusion.

Objectives:

We introduce a family of fiducial markers that is
- *informative:* exposing local deformation differentials;
- *inconspicuous:* covering product surfaces with markings, granting robustness to tampering and warp;
- *template-free:* rectifying complex deformations up to translation without phase correlation; and
- *configurable:* accomodating a variety of payloads.

1.1. Existing markers are unsuitable for traceability

Our area of focus–anti-diversion traceability–involves smaller payloads than typical 2D communication schemes (on the order of 8 message bytes, or 10^{19} messages), but much larger payloads than typical marker ID dictionaries. Even the standard "low payload" 25×25 module QR code version 2 carries 47 bytes, of which 19-37 are message bytes, depending on the error-correction-level setting [26]. This is far more than we require. On the other hand, marker systems for augmented reality applications, such as ARToolKit [10], AprilTags [38], and ArUco markers [19], contain IDs in a pre-defined dictionary, typically with on the order of 10^1 to 10^3 IDs per dictionary–far less than an eight-byte message (10^{19}). While CALTags could offer such combinatorial possibilities, they do not incorporate forward error correction (FEC); ARTags, which do, accommodate only 10^3 marker IDs.

Most fiducial markers have sufficient landmarks and asymmetry to rectify homographies. (With information to estimate normal translations, such as camera intrinsics, these markers–especially AprilTags and ArUco markers–are used by the robotics community to estimate the six degree-of-freedom pose of a camera relative to a planar surface [22, 27].) In the literature, however, only marker boards can rectify complex planar deformations induced by the acquisition of a 2D code printed on a complex surface or by camera lens warp. But these conspicuous marker boards, while useful for instrument calibration, are ill-suited for use in commercial packaging.

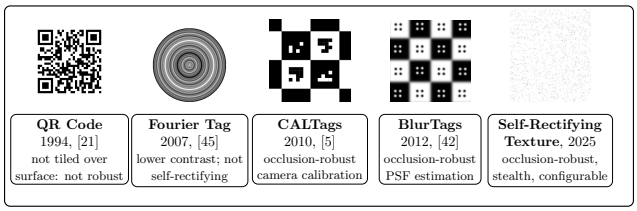


Figure 2. Example fiducial markers. The self-rectifying texture (right) is more discreet than the others in the literature. Most tags with sufficient asymmetry (all but Fourier Tag here) possess enough information to perform landmark-based rectification of a homography, but only conspicuous chessboard patterns repeated across a surface or the self-rectifying texture can reveal the more complex planar deformations encountered in product traceability applications and exhibit the necessary robustness to tampering. Like the QR code, it can be configured to carry arbitrary messages (rather than a fixed set of motifs from a small marker dictionary); its payload size, density, and robustness are highly adaptable, akin to QR code version and error-correction-level selection.

While many markers offer some protection against occlusion, only a furtive marker that can cover a large surface is robust to deidentification attacks.

Traceability applications have varying two-dimensional code design needs. For codes with fine-grained detail, often examined with specialized instruments at high levels of magnification, small contiguous regions must be decodable. For codes masked by branding materials and product information and acquired with a smartphone, a code block can span a large surface, with the parity-check matrix linking spatially distant bits. Another approach suitable for commercial packaging consists in printing random linear combinations of small blocks–small enough to fit between branded product markings–so that the complete message can be decoded with enough acquired linear combinations.

1.2. Direct rectification of planar markers

The perspective projection of a 3D scene onto an image plane induces geometric distortion. Information displayed on a planar surface—such as a QR code on the face of a cuboid package–that is acquired with perspective can be placed in a standard view (fronto-parallel, typically) via a homography. A homography $\mathrm{H} : \mathbb{R}^2 \to \mathbb{R}^2$ is a nonlinear planar deformation determined by an isomorphism of the real projective plane $\mathbb{P}_2$ (the set of lines through the origin in $\mathbb{R}^3$). Thus, each homography maps distinct lines to distinct lines, preserving incidence. While homographies preserve collinearity, they break parallelism and distort angle and distance ratios. This distorts encoded data motifs, rendering their accurate recovery difficult. The rectifying homography is found using the aquired pixel coordinates of

at least four spatial landmarks [22], whose locations in a rectified code are generally known. Applying the rectifying homography to the acquired image places the data in a standard form accessible to the decoder.

But spatial landmarks impede stealth: subtle landmark motifs are difficult to detect reliably. One can compensate for poor keypoint detection reliability with quantity. With knowledge of the entire fiducial marker that was acquired in standard form (which is rare in traceability applications), one can use algorithms such as SIFT to find available keypoint correspondences between the known and acquired patterns, from which the rectifying homography for the acquired image can be calculated [22, 33]. But such an approach requires explicit knowledge of the entire pattern (rather than a small set of reliably detected, high-contrast landmarks, like QR code "eyes," which do not contain variable coded data and thus can be known without a complete template). Even with template access, such approaches may still suffer from matched descriptor points that are degenerate, unreliable, or unavailable without tweaks to the keypoint isolation algorithm. This especially holds for patterns made from faint, repetitive motifs, such as the patterns we seek to create (the ideal traceability markers are imperceptible). For traceability markers, one often has access to the *sites* at which data might be encoded but not the data itself. In such cases, cross-correlation-based methods are more practical than correspondence-based methods. Such methods are computationally expensive and ill-suited for the recovery of many rectifying homography parameters. An overview of existing inverse, template-free rectification methods is provided in the Supplementary Materials, Sec. A.

1.3. Self-rectifiable textures: textures that pose a natural inverse problem for rectification

Fiducial markers are examples of *self-rectifiable patterns*. Structures embedded within the marker–typically the pixel indices of landmarks–reveal information about the pose of the acquisition camera. These patterns may possess data payloads (channel coded, for robustness to occlusion [21]) or auxiliary calibration tools (such as dots for the identification of spatially varying point-spread functions [43]). They announce themselves to the acquisition camera with large, elementary geometric forms and high-contrast, typically binary, signals. Notable exceptions are the ARToolKit [39] and Fourier tag [46] (see Fig. 2). The former contains a large square image that is the weighted sum of a payload consisting of a Discrete Cosine Transform (DCT-II) basis function (or its negative) and a fixed DCT-II basis function used to compensate for the bilateral symmetry of even basis functions. Marker identification is accomplished using phase correlation. The Fourier tag embeds in a large circle a smooth radial signal, fixed across a sector. This signal is the Discrete Fourier Transform (DFT) of a symmetrized binary signal that permits partial recovery (of its low-frequency component) in the presence of blur. In both cases, the fiducial marker is grayscale and smooth.

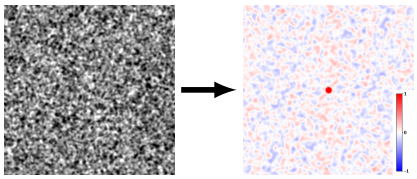
Motivated by traceability applications, we seek to design self-rectifying patterns in the form of inconspicuous textures that can be printed across a curved packaging surface. Deformed, cropped texture portions can be rectified modulo translation using a single image without template, based on known statistical properties of the texture.

To achieve inconspicuousness, we place landmarks not in the texture itself but rather in its autocorrelation image. Since the positions of peaks in the autocorrelation image are mapped linearly by the deformation differential, the autocorrelation of patches of a deformed texture expose the local behavior of the deformation. Moreover, to rectify the deformation, the original texture need not be known, only the sites of its most prominent autocorrelation peaks. Template-free rectification (modulo translations) can thus be performed, avoiding expensive phase correlations.[1] With additional knowledge of texture (such as the sites at which data appear in the binary texture in the rectified view), the translation can be acquired with a single phase correlation, thanks to the separability property described in Sec. 3. Thus, the complete affine approximation of the deformation of the patch can be acquired (and rectified) with a single phase correlation, in sharp contrast with the computationally expensive correlation-based acquisition of up to six affine parameters in the template-matching literature. Such methods require a phase correlation in each step of the optimization [28].

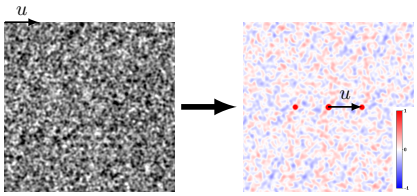
While a single square marker, such as a QR code, can only be used to rectify homographies and chessboard arrays of square markers are ill-suited for the aesthetic demands of packaging, *self-rectifiable textures*[2] can be inconspicuously placed across a curved surface, making its rectification possible and its information decodable. As these textures are repeated and contain no landmarks in the spatial domain, there is no unique site of vulnerability. Data payloads encoded in such textures can be made robust to occlusion and tampering via traditional channel coding over the whole texture, over blocks, or via random linear combinations of blocks. In this way, no particular part of the texture

[1] These have cost $O(mn \log mn)$, where m and n are the maximum dimension of the query patch and template. Thus, they are "as expensive" as an autocorrelation computation. If the template has 100 times the area of the query patch, one can compute over 100 patch autocorrelations for the cost of a phase correlation.

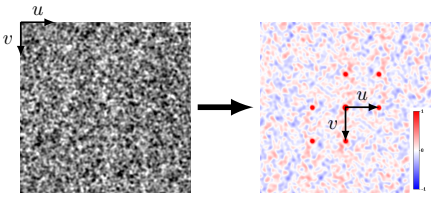
[2] This usage is unrelated to that of [54]. In that work, rectification refers to improvements in constrained synthetic texture generation; "self-rectification" hints at the self-attention in the training process, not the texture itself. In our work, we use "self-rectifying" or "self-rectifiable" to indicate that the texture itself contains sufficiently rich geometric information to enable template-free rectification of planar deformations modulo translation.



(a) A texture and its autocorrelation (most prominent peak at the origin).



(b) Two overlaid copies with pairwise lags $\pm u$; its autocorrelation.



(c) Three copies, with pairwise lags $\pm u$, $\pm v$, and $\pm(u-v)$. The autocorrelation peak locations $H_{\text{f-p}} = \{\pm u, \pm v, \pm(u-v)\}$ form the "fundamental hexagon." Affine deformation $x \mapsto \mathbf{A}x + b$ of the texture maps peak locations linearly to a deformed hexagon $\{\pm\mathbf{A}u, \pm\mathbf{A}v, \pm\mathbf{A}(u-v)\}$.

Figure 3. Three textures in fronto-parallel view (left) and their autocorrelation images (right): a base texture f in (a); $f + f(\cdot - u)$ in (b); and $f + f(\cdot - u) + f(\cdot - v)$ in (c). The texture of (b) is used in [42] to rectify scale and rotation without reflection. The texture in (c) has a "fundamental hexagon" of autocorrelation peaks; their locations in deformed image patches are mapped linearly by the deformation differential, permitting template-free rectification of more general deformations modulo translation.

need be acquired to decode a message, only *enough* of the texture.

1.4. Outline

In this article, we generate simple autocorrelation-based self-rectifying textures by summing three copies of a texture with noncollinear offsets (see Algorithm 1), generating a hexagon of known peaks in the autocorrelation image of the texture in the fronto-parallel view (see Fig. 3). When patches of the texture are acquired with perspective on a planar or curved surface, the hexagon in the patch autocorrelation is mapped linearly by differential of the planar deformation. The relative offsets of autocorrelation peaks in a patch of the query image expose the linear part of the deformation between the patch in fronto-parallel view and in the query view; this linear part is found in Algorithm 2. The translation component can be computed using phase correlation (with knowledge of the texture or just locations where data motifs in the texture might be present), enabling

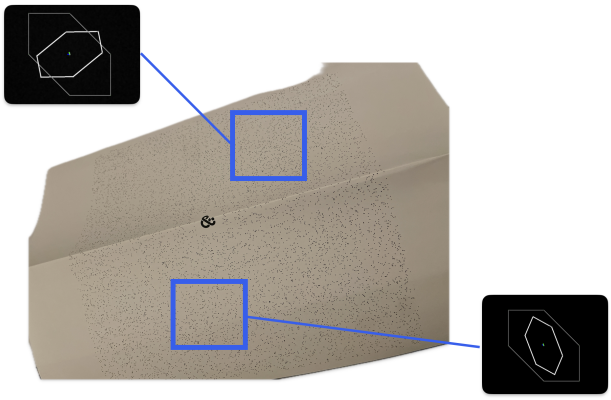


Figure 4. A self-rectifiable texture covering a rectangular prism and the fundamental hexagons of two patches. The orientation of the autocorrelation peaks of each patch reveals the differential of the texture deformation at the patch.

patchwise rectification of complex deformations, as in Algorithm 3. Textures are generated in Sec. 2 and affine maps are rectified in Sec. 3. The robustness of patch-based local affine map acquisition is assessed in Sec. 4. Fully template-free rectification of deformations modulo translations is considered in Sec. 5 and the Supplementary Materials, Sec. C.

2. Generation of self-rectifiable textures

Fig. 3 illustrates the transformation of a sample texture (in fronto-parallel view) into one that is self-rectifying. We require a base texture, such as that of Fig. 3a, whose most prominent autocorrelation peak is at the origin. The autocorrelation of two overlaid copies of the texture in Fig. 3b, at relative shifts 0 and u, possesses three prominent peaks, at 0 and $\pm u$. These peak locations are used to rectify rotation-dilations (orientation-preserving similarities with at least one fixed point) in [42].

A self-rectifying texture is presented in Fig. 3c. Created using Algorithm 1, this texture consists of three summed copies of the base texture with relative shifts of 0, u and v. This shifted overlay results in a "fundamental hexagon" of its most prominent autocorrelation peaks at six nonzero lags: $H = \{\pm(u-0), \pm(v-0),$ and $\pm(v-u)\}$. Under mild assumptions, the fundamental hexagon and origin form the seven most intense peaks in the autocorrelation image; our ability to isolate them depends on the prominence of the peak at the origin in the autocorrelation image (and, in practice, the patch size). In the ideal case, the self-rectifying texture consists of three summed, shifted copies of white noise, with autocorrelation equal to $3\delta_0 + \sum_{p\in H}^{6} \delta_p$ (where δ_p is the shifted Dirac delta $\delta(\cdot - p)$). But this works far beyond the ideal case: three summed, shifted copies of even substantially self-correlated textures have recoverable

fundamental hexagons.

From these six peak locations–all samples of the oblique Bravais lattice [24] with primitive vectors u and v–we can extract the transformed shifts u and v. For instance, taking all sums and differences of two distinct vectors and reflected copies $\pm a$ and $\pm b$, we recover the third peak and the vector sum of two peaks only when $\{\pm a, \pm b\} = \{\pm u, \pm v\}$. From the offsets alone we cannot uniquely identify u and v in general without additional information, such as their relative positions in the rectified texture and constraints on the affine transformation; oriented motifs in the base texture; additional shifted copies; or distinct weights in the sums of the shifted copies of the base texture. Of course, if we know the form of the rectified texture, we can execute the assignment between hexagon peaks and shifts using the phase correlation.

As Sec. 3 details, affine transformations $x \mapsto \mathbf{A}x + b$ of the texture image effect linear transformations of its autocorrelation, and of the positions of the autocorrelation peaks. The fundamental hexagon is transformed linearly by $\mathbf{A}$. This result extends in practice to patches of the texture, as Fig. 4 demonstrates. Thus, the peaks in the autocorrelation image of a patch of a texture expose the linear part of the local affine approximation to the deformation we wish to rectify. This approach is novel but familiar: autocorrelation peaks are widely used in signal processing, from channel estimation (*e.g.*, using m-sequences) to watermark recovery [29].

Algorithm 1: One way to create self-rectifying textures: superimpose three shifted copies of a texture.

```
// Key property: if u and v are not collinear
   and f has its most prominent autocorrelation
   peak at 0, the most prominent autocorrelation
   peaks of r form the ``fundamental hexagon.''
```

Input: texture $f : \mathbb{R}^2 \to \mathbb{R}$; shifts u and v in $\mathbb{R}^2$

Result: $r = f + f(\cdot - u) + f(\cdot - v)$

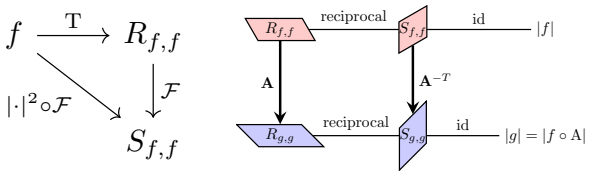
Algorithm 1 is but one of many ways to create self-rectifiable textures. It is the simplest method–used here to streamline the exposition and keep focus on the stealth nature of self-rectifying textures, which consequently require inverse rather than correspondence-based rectification–but it is not the one we use in practice. One can place fixed constellations of points around repulsively sampled points. More shifts can be employed. One can choose instead to place peaks in the power spectrum instead (which slightly changes the rectification problem; see Fig. 5a). More generally, one can search for textures with desired autocorrelation (or, equivalently, power spectrum) motifs. The problem of inferring a function from its autocorrelation is widely studied under the name "phase retrieval" as the autocorrelation of a signal determines its Fourier transform modulus but not phase. Classically, it arises when employing diffracted electromagnetic radiation to determine the structure of an object (*e.g.*, in X-ray crystallography, transmission electron microscopy, etc.) [45]: in such cases, one seeks to image an object with access only to (the Fourier transform of) its autocorrelation. Today, the problem is found in many other domains [3]. Homometry makes this inverse problem ill-posed (a rich space of measures, stochastic and deterministic, can possess the same Fourier transform modulus); solutions return one of many functions consistent with the known autocorrelation (see, *e.g.*, Wrinch's algorithm [47]). Unique recovery is possible only in certain constrained settings–most notably, binary images of compact convex planar shapes [6]. In our application, unlike in computational imaging, this ill-posedness is an asset, offering design flexibility to satisfy other constraints (due to printing technology, aesthetic imperatives, the need for stochasticity [20]) or to avoid repetition. Moreover, as rectification depends only on pixel locations of autocorrelation structures such as peaks, not the autocorrelation itself, the flexibility is still greater than that found in the classical phase retrieval problem.

3. Efficient extraction of local affine maps thanks to the autocorrelation's separation of translation from the linear part

For simplicity, we consider images to be integrable functions $f : \mathbb{R}^2 \to \mathbb{R}$. We are interested in the behavior of images of self-rectifying textures under deformations of the plane. Let $\mathrm{A} \in \mathrm{Aff}(2, \mathbb{R})$ be an affine transformation $x \mapsto \mathbf{A}x + b$ parameterized by $\mathbf{A} \in \mathbb{R}^{2\times 2}$ (invertible) and $b \in \mathbb{R}^2$. We observe an image g that is f deformed by an affine map: $g = f \circ \mathrm{A}$, so $g(x) = f(\mathbf{A}x + b)$. We wish to recover A knowing f (or its properties). We denote as T the map from an image f to its deterministic autocorrelation $R_{f,f} \overset{\text{def}}{=} f \star \overleftarrow{f}$ (*i.e.*, $(\mathrm{T}f)(\tau) = \int_{\mathbb{R}^2} f(x+\tau)f(x)\,\mathrm{d}x$). Translations preserve the autocorrelation; thus, by a regular variable substitution, $\mathrm{T}(f \circ \mathrm{A}) \propto \mathrm{T}(f) \circ \mathbf{A}$ (here $\propto$ reads "is proportional to"):

$$\begin{aligned} \mathrm{T}(g)(\tau) &= \int_{\mathbb{R}^2} f(\mathbf{A}(x+\tau)+b) f(\mathbf{A}x+b)\,\mathrm{d}x \qquad (1) \\ &= \frac{1}{|\det \mathbf{A}|} \int_{\mathbb{R}^2} f(t+\mathbf{A}\tau) f(t)\,\mathrm{d}t \propto \mathrm{T}(f)(\mathbf{A}\tau). \end{aligned}$$

Our forward model can be stated simply: if p is the location of a peak in f, then $\mathbf{A}p$ is the location of a peak in $\mathrm{T}f$. The inverse problem for the recovery of the affine map's linear part $\mathbf{A}$ from the autocorrelation of $g = f \circ \mathrm{A}$ is solved in Algorithm 2. For patchwise rectification, we can recover the phase b using phase correlation. Crucially, this approach, unlike typical template-matching approaches (discussed in



(a) Wiener-Khinchin theorem. (b) Oblique lattice of peaks.

Figure 5. By the commutative diagram (a), an oblique lattice of peaks in a texture f and its power spectrum $S_{f,f}$ is reciprocal to that of its autocorrelation $R_{f,f}$. In (b), these signal peak locations are related to those of $g = f \circ \mathrm{A}$, deformed by affine map A with linear part $\mathbf{A}$. Observations of autocorrelation peaks of a deformed texture expose the linear part $\mathbf{A}$ of an affine deformation; observations of power spectrum peaks, $\mathbf{A}^{-T}$.

A), separates recovery of $\mathbf{A}$ and b so that only these two parameters need be recovered via phase correlation.

Define the operator $\mathrm{N}_g(f)$ giving the phase correlation with an image g of an image f as follows:

$$\mathrm{N}_g f = \mathcal{F}^{-1}\left(\frac{\widehat{f}(\omega)\overline{\widehat{g}(\omega)}}{|\widehat{f}(\omega)\overline{\widehat{g}(\omega)}|}\right).$$

By a regular variable substitution,

$$\widehat{f \circ \mathrm{A}}(\xi) = \frac{1}{|\det \mathbf{A}|} e^{ib^T \mathbf{A}^{-T} \xi} \widehat{f}(\mathbf{A}^{-T}\xi). \tag{2}$$

From (2), we can establish the translation-covariance of the operator N_f: any pure translation $\mathrm{A}_t : x \mapsto \mathbf{I}x + b$ satisfies

$$\mathrm{N}_g(f \circ \mathrm{A}_t) = \mathrm{N}_g(f) \circ \mathrm{A}_t.$$

As translations do not affect the autocorrelation, we can rectify $\mathbf{A}$ and b separately, as in Algorithm 3. Due to the robustness of the phase correlation, patchwise rectification is often possible when one lacks access to a complete template consisting of the fronto-parallel view of the self-rectifiable texture, but does have the *sites* where coded data may appear (say, in a binary image where only pixels that communicate data may be illuminated).

Eq. (2) allows us to complete Fig. 5b, which shows how peak locations in the Fourier transform magnitude, power spectrum, or autocorrelation of a Bravais lattice are transformed linearly when the lattice image undergoes an affine deformation A. Our example self-rectifying texture, generated using Algorithm 1, possesses not a lattice of peaks in the autocorrelation domain, but seven samples thereon (the fundamental hexagon). This windowing renders the dual-lattice peaks in the power spectrum too difficult to detect to permit an estimation of $\mathbf{A}$. However, the power spectrum peaks of other self-rectifying textures may be used to estimate the linear part of the affine deformation $\mathbf{A}$.

Algorithm 2: Estimate linear component of affine transformation from fundamental hexagon.

Input: fundamental shifts u and v in $\mathbb{R}^2$; patch P of deformed texture

Result: linear part $\mathbf{A}$ of affine deformation

`// Compute periodic component [32] autocorrelation`

$R = \texttt{idft2}(|\texttt{dft2}(\texttt{per\_component}(P))|^2)$

`// if most prominent peaks exhibit the correct form, fit a Gaussian to each peak's neighborhood, give subpixel maxima locations`

$H_{\text{def}} = \texttt{get\_fundamental\_hexagon}(R)$

`// solve the assignment problem between peaks in` $H_{\text{f-p}} = \{\pm u, \pm v, \pm(u-v)\}$ `and their deformed twins in` H_{def} `using oriented texture motifs, weights in the overlay sum, constraints on` $\mathbf{A}$ `and identification of lattice primitive vectors, or phase correlation`

$\texttt{match} \leftarrow \texttt{solve\_assignment}(\mathbf{A}H_{\text{f-p}}, H_{\text{def}})$

`// A is least-squares fit to an overdetermined system, well-posed whenever` u, v `noncollinear`

$$\mathbf{A} = \underset{\mathbf{A}\in\mathbb{R}^{2\times 2}}{\operatorname{argmin}} \sum_{i=1}^{6} ||\mathbf{A}H_{\text{f-p}}[i] - H_{\text{def}}[\texttt{match}(i)]||_{\mathbb{R}^2}^2 \tag{3}$$

4. Robustness of patchwise rectification

Complex deformations are often rectified patchwise, by fitting affine maps or homographies to each deformed patch. These maps are widely used in remote sensing, image stitching, and registration applications, as well as–in reverse–in image projection on nonplanar surfaces [4, 31, 55]. Algorithm 3 details our patchwise rectification process. Fig. 6 shows patchwise rectification of a homography.

The change of variables used in Eq. (1) assumes an infinite texture. Preprocessing can enhance peak prominence in finite textures, particularly for small or axis-aligned shifts. In Algorithm 2, we retain patch periodic components [32]. Texture motif and density choices also impact hexagon peak acquisition. In this study, we consider two base textures: a random binary texture and a binary commercial marker of 3×3 square motifs placed randomly on a regular grid of a tunable density, using a (128, 64) low-density parity check (LDPC) code. The regular grid of the latter sacrifices peak prominence for ease of decoding.

Patch sampling strategies require great care when deformations are not affine. Patches barely larger than the fundamental shifts u and v lack signal correlation runtime; the echoes of shifted copies are barely recognizable, and the corresponding autocorrelation peaks are hard to dis-

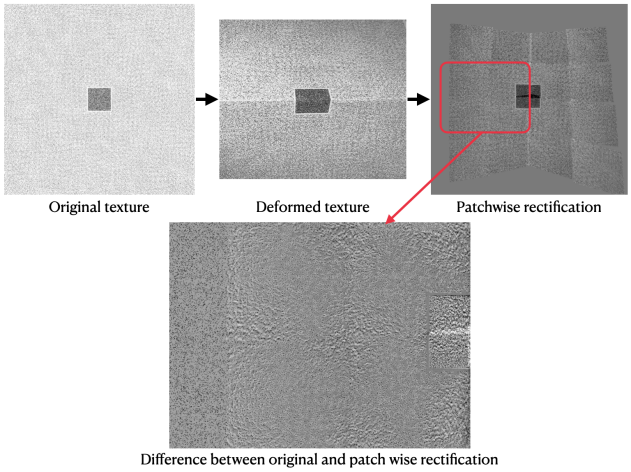


Figure 6. Patchwise rectification of a self-rectifying texture printed at 300 DPI and affixed across a table border. Patchwise rectification is accomplished using Algorithm 3, given knowledge of the *sites* at which data may be encoded in a commercial traceability marker but not the template code. As the difference image attests, redressed points are sufficiently well-placed to enable decoding.

tinguish from the background. Moreover, due to the use of the DFT to compute the autocorrelation, their locations can be aliased (see Supplementary Materials, Fig. 7). With patch lengths a bit over twice the fundamental shifts, peaks are prominent. Beyond that, our linearized forward model can break down: peaks of large patches appear at locations inconsistent with a deformed fundamental hexagon $\mathbf{A}H_{\text{f-p}}$ due to approximation error. Affine deformations map peaks situated on an oblique Bravais lattice in $H_{\text{f-p}}$ to peaks on another oblique lattice $\mathbf{A}H_{\text{f-p}} = H_{\text{def}}$; non-affine maps warp the lattice. While the difference between primitive vectors is in the fundamental hexagon $u - v \in H_{\text{f-p}}$, this does not hold for their deformed pairs u' and v': $(u' - v') \notin H_{\text{def}}$. We consider the hexagon H_{def} undetectable when we cannot isolate the primitive vectors of H_{def} in this way, within tolerance.

In the Supplementary Materials, Sec. D, we show that the recovery of the linear part $\mathbf{A}$ of an affine map from the fundamental hexagon is possible even with poor conditioning. It is standard [22] to write the singular value decomposition of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = (\mathbf{U}\mathbf{V}^T)\mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$ in the orientation-preserving subgroup of $\mathrm{GL}(2,\mathbb{R})$ as a rotation $\mathbf{V}^T$ by a "tilt angle" ϕ, a non-isotropic scaling by a diagonal matrix $\mathbf{D} = \mathrm{diag}(1,t)$ in the rotated coordinates, a rotation $\mathbf{V}$ by $-\phi$, a final rotation $\mathbf{U}\mathbf{V}^T$ by θ, and an isotropic scaling (zoom) λ. When $t < 1$, the tilt t is the inverse of the condition number of the linear part $\mathbf{A}$ of an affine map.

While a continuous image of an infinite texture always permits recovery of nondegenerate $\mathbf{A}$, per Equation (1), reliable recovery of the transformed hexagon from a finite, sampled patch depends on the texture, sampling strategy (algorithm to choose patch size(s), locations, and number), and acquisition mode. These parameters must be assessed with experimentation specific to the application. In our experiments, we used the naive strategy of sampling patches of a fixed size uniformly for piecewise rectification and randomly, with variable size determined by Supplementary Materials Algorithm 6, for homographic rectification.

Algorithm 3: Patchwise rectification of a planar deformation. Requires phase correlation and a template (fronto-parallel view of texture or simply the *sites* at which binary data may be encoded).

Input: A template T and deformed texture I
Result: Patchwise rectified image R
// I is divided into patches, overlapping where needed, with size and shape chosen so that the periodic component of each patch has a detectable transformed fundamental hexagon.
$P_1, \ldots, P_n = \texttt{patchify}(I)$
// Get the local affine approximation of the deformation φ. For each patch P_i centered at y_i, our estimate of the deformation's differential $\nabla_{\varphi^{-1}(y_i)}\varphi$ is $\mathbf{A}_i$ and with the phase b_i we complete the first-order approximation about $x_i = \varphi^{-1}(y_i) = \mathbf{A}_i^{-1}(y_i - b_i)$: $h \mapsto y_i + \mathbf{A}_i(h - x_i) = \mathbf{A}_i h + b_i$.
for $i = 1, \ldots, n$ **do**
  $\mathbf{A}_i = \texttt{get\_linear\_component}(P_i)$
  $b_i = \operatorname{argmax}_{b \in \mathbb{R}^2} \mathrm{N}_{\mathbf{A}_i^{-1} P_i}(b)$
// Apply the inverse map and interpolate.
for $i = 1, \ldots, n$ **do**
  Form R_i by applying the map $h \mapsto \mathbf{A}_i^{-1}(h - b_i)$ to P_i, using bicubic interpolation to remain on pixel grid
Stitch together the overlapping P_i to form P.

5. Extension to template-free rectification

The patchwise rectification described in Algorithm 3 requires access to a template: an image in the fronto-parallel view of the binary texture being rectified or of sites where data may appear. This template is used to recover the translation component of the local affine map via a phase correlation. In certain applications, rectification of a deformation without access to such a tempalte is desirable.

From the autocorrelation peaks of each deformed patch P_i, centered at y_i, we can estimate the differential

$$\mathbf{A}_i \approx \nabla_{\varphi^{-1}(y_i)}\varphi \tag{4}$$

of the deformation φ using Algorithm 2. For this, we need

no template image, only the shift parameters u, v used to generate the self-rectifiable texture using Algorithm 1 and modest additional information to distinguish symmetries and resolve the assignment problem between peaks. The differential of a homography, for instance, involves all its parameters, including the translation parameters. It is natural, then, to wonder whether φ can be recovered from our indirect observations $\mathbf{A}_i$ of its differential $\nabla_{\varphi^{-1}(y_i)}\varphi$, made using peak displacement.

Not entirely, it turns out. As our observations are made through the translation-invariant autocorrelation, they cannot distinguish between two deformations related by translation, φ and $\varphi \circ \tau$. By the chain rule, the local affine transformations estimated using P_i, centered at y_i, of both φ and $\varphi \circ \tau$ share a linear part, since $\nabla_x \tau = \mathbf{I}$ for all $x \in \mathbb{R}^2$:

$$\begin{aligned}\nabla_{(\varphi\circ\tau)^{-1}(y_i)}\varphi \circ \tau &= \nabla_{(\tau\circ\tau^{-1}\circ\varphi^{-1})(y_i)}\varphi \cdot \nabla_{(\varphi\circ\tau)^{-1}(y_i)}\tau \\ &= \nabla_{\varphi^{-1}(y_i)}\varphi \cdot \mathbf{I} = \nabla_{\varphi^{-1}(y_i)}\varphi.\end{aligned}$$

Thus, our indirect observations alone cannot be used to distinguish φ and $\varphi \circ \tau$; we can only hope to recover φ modulo translation. Nevertheless, we can still rectify without template by assembling our scattered observations $\mathbf{A}_i$ into a global model of φ modulo translation. With access to our deformation modulo translation, we need just a single observation of the translation–say, via a synchronization sequence, or via information encoded in our texture–to resolve the ill-posedness of our inverse problem and recover φ. Moreover, in certain circumstances, recovering φ modulo translations suffices to decode the message.

To assemble our scattered observations $\mathbf{A}_i$ of $\nabla_{\varphi^{-1}(y_i)}\varphi$ into an estimate of φ modulo translations, we have two approaches: by fitting the $\mathbf{A}_i$ to a parametric model of φ or by obtaining a non-parametric representation of φ.

Template-free rectification can be performed nonparametrically, by integrating the observations (Eq. (4)) directly over a triangle mesh into a global model. The two rows of our observations $\mathbf{A}_i$ form an integrable polyvector field. The literature on polyvector fields [9, 41, 51] offers ways to integrate [15] these observations. They also permit denoising: a polyvector field can be represented as a polynomial, determined by two complex coefficients, that exhibits the same symmetries [40]. The Dirichlet energy of the vector field of coefficients across the image or graph of sample sites is widely used as a smoothness penalty.

In practice, parametric methods require fewer observations to characterize the rectifying map for a marker on a packaging surface and thus are more suitable for real-time rectification than direct integration. We can model the *rectifying* deformation parametrically–using, for instance, homographies, polynomials, or thin-plate splines–and fit its parameters to our inverted observations $\mathbf{B}_i = \mathbf{A}_i^{-1}$. In the Supplementary Materials, Sec. C, we show that the problem of fitting rectifying homographies modulo translation to these observations can be reduced to an optimization over a six-dimensional affine subspace of $\mathbb{R}^9$. In short, we can view our homographies–the projective linear group $\mathrm{PGL}(3,\mathbb{R})$–as isomorphic to the special linear group $\mathrm{SL}(3,\mathbb{R})$ as a Lie group and manifold. The quotient manifold of $\mathrm{SL}(3,\mathbb{R})$ modulo translations is covered by the union of three manifolds. Only one of these manifolds, the Lie group $\mathrm{A}_2(2,\mathbb{R})$ of matrix transposes of the affine group [13], is relevant to the rectification of homographies encountered in practice. The manifold $\mathrm{A}_2(2,\mathbb{R})$ is dense in a six-dimensional affine subspace $\mathcal{H}$ of $\mathbb{R}^9$ over which we optimize, using `pymanopt`'s trust-region algorithm [1, 50] and Euclidean manifold. As we do not encounter singular optima in practice, we can interpret the result as a member of $\mathrm{A}_2(2,\mathbb{R})$, representing an orbit of $\mathrm{SL}(3,\mathbb{R})$ modulo translations–the matrix that has been "translated to zero."

Let H be the rectifying homography and $\mathrm{G} = \mathrm{H}^{-1}$ the deformation. The scattered observations $\mathbf{A}_i$ of $\nabla_{\mathrm{G}^{-1}(y_i)}\mathrm{G}$ computed with Algorithm 2 can be used to estimate H. By the inverse function theorem, $\mathbf{B}_i = \mathbf{A}_i^{-1}$ is an estimate of $\nabla_{y_i}\mathrm{H}$. Using the robust (non-squared) Frobenius norm loss, we choose parameters of H by matching the parametric form $\nabla_{y_i}\mathrm{H}$ to our scattered data $\mathbf{B}_i = \mathbf{A}_i^{-1}$ in Algorithm 4.

Algorithm 4: Estimation of homography.

Input: Observations $\{\mathbf{A}_i\}_{i=1}^n$ of $\{\nabla_{\mathrm{G}^{-1}(y_i)}\mathrm{G}\}_{i=1}^n$

Output: Rectifying homography H

Solve $\mathrm{H}^* = \underset{\mathrm{H}\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^n ||\nabla_{y_i}\mathrm{H} - \mathbf{A}_i^{-1}||_F$

Nonparametric homography recovery is demonstrated in Supplementary Materials, Fig. 9c, Fig. 8, and Sec. E.

6. Conclusion

Self-rectifying textures permit the estimation of local surface geometry and perspective using a single image without keypoint matches. Rectification can be performed patchwise, using correlation with texture templates (or data encoding sites), or via model-fitting. Only the two translation parameters are fit with expensive phase correlations; the texture exposes the other four for affine patchwise rectification or homographic rectification modulo translation. With no spatial landmarks, these stealth textures can be placed across a surface, granting robustness to tampering. Since rectification depends on the autocorrelation, it is robust to a variety of contaminants that preserve locations of prominent peaks, such as blur and noise. Acquisition of the linear component of the local affine approximation to a deformation is robust and accurate in traceability applications.

References

[1] P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on Riemannian manifolds. *Found. Comp. Math.*, 7:303–330, 2007. 8

[2] Dror Aiger, Daniel Cohen-Or, and Niloy J Mitra. Repetition maximization based texture rectification. In *Computer Graphics Forum*, pages 439–448. Wiley Online Library, 2012. 11

[3] Emmanuel Amiot. Homometry and the phase retrieval problem. *Music Through Fourier Space: Discrete Fourier Transform in Music Theory*, pages 27–49, 2016. 5

[4] Vicente Arevalo and Javier Gonzalez. Improving piecewise linear registration of high-resolution satellite images through mesh optimization. *IEEE Transactions on Geoscience and Remote Sensing*, 46(11):3792–3803, 2008. 6

[5] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. Caltag: High precision fiducial markers for camera calibration. In *Vision, Model., Visualization*, pages 41–48, 2010. 1

[6] Gennadiy Averkov and Gabriele Bianchi. Confirmation of Matheron's conjecture on the covariogram of a planar convex body. *Eur. J. Math.*, 11(6):1187, 2009. 5

[7] Coloma Ballester and Manuel González. Affine invariant texture segmentation and shape from texture by variational methods. *J. Math. Imaging Vis.*, 9:141–171, 1998. 11

[8] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002. 11

[9] Mikhail Bessmeltsev and Justin Solomon. Vectorization of line drawings via polyvector fields. *ACM Trans. Gr.*, 38(1): 1–12, 2019. 8

[10] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. Artoolkit: A computer vision based augmented reality toolkit. *IEEE VR2000, New Jersey*, 2000. 2

[11] Jacques Brochard, Majdi Khoudeir, and Bertrand Augereau. Invariant feature extraction for 3D texture analysis using the autocorrelation function. *Pattern Recogn. Lett.*, 22(6):759–768, 2001. 11

[12] Vicent Caselles, Bartomeu Coll, and Jean-Michel Morel. Geometry and color in natural images. *J. Math. Imaging Vis.*, 16:89–105, 2002. 11

[13] Yevhenii Yu Chapovskyi, Serhii D Koval, and Olha Zhur. Subalgebras of Lie algebras. Example of $\mathfrak{sl}(3,\mathbb{R})$ revisited. *arXiv preprint. arXiv:2403.02554*, 2024. 8, 14

[14] U.S. Consumer Product Safety Commission. Tracking label business guidance. Technical report, Guidance (updated October, 2022), 2022. 1

[15] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. Integrable polyvector fields. *ACM Trans. Graph. (TOG)*, 34(4):1–12, 2015. 8

[16] Csaba Domokos, Jozsef Nemeth, and Zoltan Kato. Nonlinear shape registration without correspondences. *IEEE Transactions on pattern analysis and machine intelligence*, 34(5): 943–958, 2011. 11

[17] Mark Fiala. ARTag, a fiducial marker system using digital techniques. In *CVPR 2005: IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, pages 590–596. IEEE, 2005. 1

[18] Directorate-General for Environment. *Proposal for Ecodesign for Sustainable Products Regulation*. The European Commission, 2022. 1

[19] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.*, 47(6):2280–2292, 2014. 2

[20] Uwe Grimm and Michael Baake. Homometric point sets and inverse problems. *Z. Kristallogr.*, 223(11-12):777–781, 2008. 5

[21] Masahiro Hara, Motoaki Watabe, Tadao Nojiri, Takayuki Nagaya, and Yuji Uchiyama. Optically readable two-dimensional code and method and apparatus using the same, 1998. US Patent 5,726,435. 1, 3

[22] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003. 2, 3, 7

[23] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2 edition, 2004. 15

[24] Yuchen He and Sung H Kang. Lattice identification and separation: theory and algorithm. *SIAM J. Imaging Sci.*, 12(4): 2063–2096, 2019. 5

[25] Berthold KP Horn and Michael J Brooks. The variational approach to shape from shading. *Comput. Gr. Image Process.*, 33(2):174–208, 1986. 11

[26] International Organization for Standardization and International Electrotechnical Commission. Information technology – automatic identification and data capture techniques – qr code bar code symbology specification, 2015. Standard number: ISO/IEC 18004:2015. 2

[27] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *Journal of Intelligent & Robotic Systems*, 101(4):71, 2021. 2

[28] Shun'ichi Kaneko, Yutaka Satoh, and Satoru Igarashi. Using selective correlation coefficient for robust image registration. *Pattern Recognition*, 36(5):1165–1173, 2003. 3, 11

[29] Martin Kutter. Watermarking resistance to translation, rotation, and scaling. In *Multimedia Syst. Appl.*, pages 423–431. SPIE, 1999. 5

[30] Jongwoo Lim and Ming-Hsuan Yang. A direct method for modeling non-rigid motion with thin plate spline. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 1196–1202. IEEE, 2005. 11

[31] Chung-Ching Lin, Sharathchandra U Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y Aravkin. Adaptive as-natural-as-possible image stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1155–1163, 2015. 6

[32] Lionel Moisan. Periodic plus smooth image decomposition. *J. Math. Imaging Vis.*, 39:161–179, 2011. 6

[33] Lionel Moisan, Pierre Moulon, and Pascal Monasse. Fundamental matrix of a stereo pair, with a contrario elimination of outliers. *Image Process. Line*, 6:89–113, 2016. 3

[34] Mads Nielsen, Peter Johansen, Andrew D Jackson, Benny Lautrup, and Søren Hauberg. Brownian warps for non-rigid registration. *Journal of Mathematical Imaging and Vision*, 31(2):221–231, 2008. 11

[35] Jingyi Ning, Lei Xie, Yi Li, Yingying Chen, Yanling Bu, Baoliu Ye, and Sanglu Lu. Moirépose: ultra high precision camera-to-screen pose estimation based on moiré pattern. In *Proceedings of the 28th annual international conference on mobile computing and networking*, pages 106–119, 2022. 11

[36] Richard Nock and Frank Nielsen. Statistical region merging. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11):1452–1458, 2004. 11

[37] U.S. Department of Homeland Security Office of Strategy, Policy, and Plans. Combating trafficking in counterfeit and pirated goods. Technical report, U.S. Department of Homeland Security, 2020. Published pursuant to the Presidential Memorandum dated April 3, 2019. 1

[38] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011. 2

[39] Charles B Owen, Fan Xiao, and Paul Middlin. What is the best fiducial? In *IEEE Int. Workshop Augmen. Reality Toolkit*, pages 8–15. IEEE, 2002. 3

[40] David Palmer, David Bommes, and Justin Solomon. Algebraic representations for volumetric frame fields. *ACM Trans. Gr.*, 39(2):1–17, 2020. 8

[41] D. Palmer, O. Stein, and J. Solomon. Frame field operators. *Comput. Gr. Forum*, 40(5):231–245, 2021. 8

[42] Justin Picard, Jean-Pierre Massicot, Alain Foucou, and Zbigniew Sagan. Method and device for securing documents, United States Patent 0220364A1, Sep. 2, 2010. 4, 11

[43] Alexander Reuter, Hans-Peter Seidel, and Ivo Ihrke. BlurTags: spatially varying PSF estimation with out-of-focus patterns. In *WSCG 2012: Int. Conf. Comput. Gr., Visualization, Comput. Vision*, pages 239–247, 2012. 1, 3

[44] Mariano Rodríguez, Gabriele Facciolo, and Jean-Michel Morel. Robust homography estimation from local affine maps. *Image Process. Line*, 13:65–89, 2023. 14

[45] Joseph Rosenblatt. Phase retrieval. *Commun. Math. Phys.*, 95(3):317–343, 1984. 5

[46] Junaed Sattar, Eric Bourque, Philippe Giguere, and Gregory Dudek. Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *CRV 2007: Can. Conf. Comput. Rob. Vision*, pages 165–174. IEEE, 2007. 1, 3

[47] Marjorie Senechal. A point set puzzle revisited. *Eur. J. Combinatorics*, 29(8):1933–1944, 2008. 5

[48] J.S. Seo, J. Haitsma, T. Kalker, and C.D. Yoo. Affine transform resilient image fingerprinting. In *ICASSP 2003: IEEE Int. Conf. Acoust. Speech Signal Process.*, pages III–61, 2003. 11

[49] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, London, UK, 2010. 16

[50] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *J. Mach. Learn. Res.*, 17 (137):1–5, 2016. 8

[51] Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. Directional field synthesis, design, and processing. *Comput. Gr. Forum*, 35(2):545–572, 2016. 8

[52] Dor Verbin and Todd Zickler. Toward a universal model for shape from texture. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn.*, pages 422–430, 2020. 11

[53] Andrej Zdešar, Igor Škrjanc, and Gregor Klančar. Homography estimation from circular motion for use in visual control. *Robotics and autonomous systems*, 62(10):1486–1496, 2014. 11

[54] Yang Zhou, Rongjun Xiao, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Generating non-stationary textures using self-rectification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7767–7776, 2024. 3

[55] Bo Zhu, Li-jun Xie, Qi-hui Wang, Ting-jun Yang, and Yao Zheng. An intelligent projection system adapted to arbitrary surfaces. In *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pages 293–298. IEEE, 2011. 6

Supplementary Materials

A. Related inverse rectification approaches

While related to classic problems in computer vision (such as connecting geometry to properties of natural images including texture, shading, and color [7, 12, 25, 52]) and image processing (such as image fingerprinting and texture analysis [11, 48]), self-rectifying textures are distinct in that they are introduced to the scene specifically to orient downstream processing tasks in communication and geometry.

The literature contains several methods that rectify an image without using explicit landmark coordinates, by solving inverse problems to find the rectifying affine map or homography most coherent with local observations of the deformation. These observations can consist of deformations of known shapes in an image [16, 53]. Each shape in the image whose size and orientation in the fronto-parallel view is known *a priori*–but whose position is not necessarily known–reveals the local behavior of the deformation. Shapes sufficiently small can be profitably modeled as having been mapped linearly by the differential of the deformation. These affine maps can be estimated via pixel similarities matched via a solver for the assignment problem [8] or by considering cross-sectional areas and orientations of binarized forms [16], as area is scaled by the determinant magnitude of the differential's matrix representation (*i.e.*, the Jacobian). Non-rigid warps with an affine component, such as thin-plate splines, may also be used [8, 30, 34].

The template-matching literature contains some methods that rectify (modest) deformations without structure like landmarks or geometric forms fixed in advance [28]. They need not require a complete fronto-parallel image, just something that correlates well with it (such as data coding sites). The computational expense of the phase correlation, on (variants of) which such methods typically rely, limits the class of deformations that can be rectified. In our work, by contrast, we always separate out the fitting of translation parameters, so phase correlation is never used to learn more than these two parameters.

Self-rectifying textures exploit knowledge of how planar deformations displace autocorrelation peaks to formulate inverse problems that rectify planar deformations related to acquisition and surface geometry. Other properties of specific textures can be exploited for tasks closely related to rectification. Ning *et al*. [35] solve an inverse problem based on the spacing and orientation of moiré patterns in images of screens. Moiré pattern spacing can (within a given range, provided screen and camera parameters are known) reveal the distance to the pattern. Local moiré pattern orientation can divulge some information about the other pose parameters. A pose consistent with the ensemble of observed moiré patterns is found. However, these nuisance patterns appear only in limited circumstances (*e.g.*, photographs of electronic displays) and it is unclear why one would wish to rectify them. Aigar *et al*. [2] recover four parameters of a homography via an iterative approach. Assuming that the rectified image is a texture composed of small regions (like bricks) of the same dimension, they segment the texture using statistical region merging [36] and fit ellipses to each region. They seek a rectifying map that makes the major axes of all ellipses a fixed length d (and minor axes another fixed length, d'). First, they rectify a pure perspective transform (defined as a homography with just two parameters–the bottom-left and bottom-center entries of the homography matrix–diverging from the identity). For each major axis of the ellipses in the deformed image, the constraint that the rectified major axis have length d corresponds to a curve in $\mathbb{R}^2$. The two parameters are selected by choosing the maximum in an accumulator space for these two parameters (where the largest number of these curves intersect). The rectifying pure perspective map is applied, then the two parameters of an affine transformation available to this approach (given the symmetries of the ellipse) are fit, and the process is repeated, alternating between rectifying two pure perspective parameters and two parameters of an affine transformation modulo translations and symmetries. Segmentation with statistical region merging requires knowledge of the texture segments (*e.g.*, number of bricks) and clear boundaries between uniform texture elements. Its adaptation to stealth textures, perhaps with locally varying densities, would require a great deal of work. There are some similarities between the use of elliptical axes in [2] and the fundamental hexagon in our work, although the inverse problem we pose is different. Finally, Picard *et al*. introduced a texture that can rectify rotation and scale via autocorrelation peaks [42] (as in Fig. 3b); our work generalizes theirs and extends it to template-free rectification.

B. Subpixel refinement of peak locations and avoidance of aliased peak locations

We estimate the local approximation of the Jacobian $\mathbf{A}_i \approx \nabla_{\varphi^{-1}(y_i)}\varphi$ of a planar deformation φ using a patch centered at pixel location y_i in the image of the deformed texture and the shifts u and v that determine the fundamental hexagon in the fronto-parallel view. The smaller the patch, the more accurate the first-order Taylor approximation of the deformation, and the more the peaks in the deformed patch autocorrelation resemble a fundamental hexagon deformed by $\mathbf{A}_i$. When φ is not affine, the autocorrelation image of a large patch exhibits a nonlinear warp of the fundamental hexagon. Examples of this behavior can be seen in Fig. 7.

However, small patches possess less "runway" for shifted copies to overlap, and thus less intense linearly deformed hexagon peaks. Moreover, since we use the Discrete Fourier Transform to compute a $n \times n$ autocorrelation image of an $n \times n$ patch (and not a $(2n+1) \times (2n+1)$ autocorre-

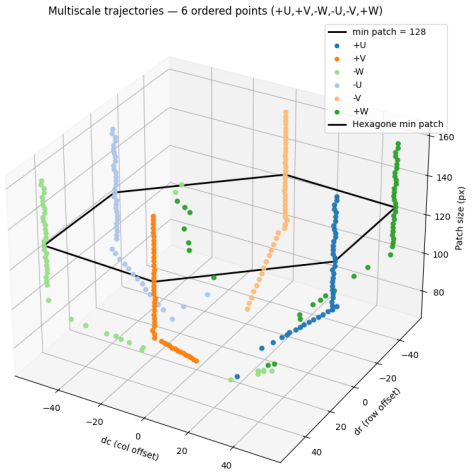


Figure 7. Patch-size selection by studying the trajectories peaks follow in the autocorrelation image as the patch size changes. When patches are too small (with dimensions of roughly twice the fundamental shift), the FFT autocorrelation calculation introduces aliasing that moves peak locations toward the center. Moreover, insufficient overlap runway may lead to insufficient peak prominence; an incorrect grid of peaks might maximize the energy E_1 (5). In large patches, the affine approximation breaks down, "smearing" or warping autocorrelation peaks, leading to optimization jitter. A good hexagon of peaks (black) is found by increasing patch size until the displacement between successive peak locations falls below a threshold (set to 0.5 pixels). This example shows the evolution of the six peak locations that form the deformed fundamental hexagon H_{def}, under a homographic deformation with inclination angles (Algorithm 7) of 20° along the x-axis and 10° along the y-axis. The stray points found under the midpoint of the edge between the light blue and light orange trajectories occur only with patch size less than 100×100 pixels (twice the dimension of fundamental shift as $||u||_{\mathbb{R}^2} = ||v||_{\mathbb{R}^2} = 50$ pixels); they appear to be found in patches of size up to 120×120 due to plotting artifacts.

lation image), peak locations can be aliased when the patch dimension n is less than twice the warped fundamental shift $\mathbf{A}_i u$ or $\mathbf{A}_i v$.

The accuracy of our estimate $\mathbf{A}_i$ of $\nabla_{\varphi^{-1}(y_i)}\varphi$ using the patch P_i centered at y_i depends on the quality of our subpixel estimates of the peak locations in the (roughly linearly) deformed fundamental hexagon H_{def}. To find $\mathbf{A}_i$ from the hexagon H_{def} estimated from P_i, we solve the inverse problem (3)

$$\mathbf{A}_i = \underset{\mathbf{A}\in \mathrm{GL}(2,\mathbb{R})}{\operatorname{argmin}} \sum_{j=1}^{6} \|\mathbf{A}H_{\text{f-p}}[j] - H_{\text{def}}[\texttt{match}(j)]\|_{\mathbb{R}^2}^2$$

using our subpixally refined measurements of peak locations in the roughly $\nabla_{\varphi^{-1}(y_i)}\varphi$-deformed hexagon H_{def} and our solution `match` to the assignment problem between the peaks in $H_{\text{f-p}}$ and H_{def}.

One way to subpixally refine the peak estimates in H_{def} is to fit a concave model (such as a Gaussian or quadratic form) to the autocorrelation image in a $p \times p$ subpatch (whose indices we place in $\boldsymbol{w}$) around the most intense autocorrelation peaks, then extract the unique maximum. We use this approach, with a spherical Gaussian model, for the affine deformations in this article. For a solution that works for a wide variety of base texture types in our self-rectifying construction (Algorithm 1), we require a more robust solution, one that works with:

- Small patches (so the lack of sufficient "overlap runway" between shifted copies can render some autocorrelation peaks less intense than those associated with other structures in the base texture); and
- Strong nonlinearities (which can "smear" or warp the peaks, spreading the intensity over many pixels).

To this end, we require a flexible interpolant and a motif shape. For the interpolant, we choose the thin-plate spline with p^2 knots corresponding to the p^2 pixels in our subpatch

$$f(x) = \alpha_1 + (\alpha_2, \alpha_3)^T x + \sum_{i=1}^{p^2} \beta_i \phi_{\text{tps}}(||x_i - x||_{\mathbb{R}^2}).$$

The weights $\alpha \in \mathbb{R}^3$ and $\beta \in \mathbb{R}^{p^2}$ are found by aligning our interpolant $f(x_i)$ with our autocorrelation pixel values $P[x_i]$ at the knot locations $\{x_i\}_{i=1}^{p^2}$ while penalizing the "wiggliness" of the interpolant. For thin-plate splines, $\phi_{\text{tps}}(x) = ||x||_{\mathbb{R}^2} \log ||x||_{\mathbb{R}^2}$. The wiggliness penalty is called the squared Beppo Levi space seminorm of order 2; it is like the usual Sobolev norm except it only penalizes the partial derivatives of order 2, not the lower-order derivatives; thus, affine functions go unpenalized. The penalty is as follows:

$$||f||^2_{\text{BL}(L^2(\mathbb{R}^2))} = \int_{\mathbb{R}^2} \left(\frac{\partial^2 f}{\partial x[0]^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x[0]x[1]^2}\right)^2 + \left(\frac{\partial^2 f}{\partial x[1]^2}\right)^2 \mathrm{d}x\,.$$

The penalty is applied over all of $\mathbb{R}^2$ (though the affine term dominates beyond our $p \times p$ window of control-point pixels). The $n + 3$ parameters $\alpha_1, \alpha_2, \alpha_3, \beta_1, \ldots, \beta_n$ are found by solving a matrix equation that identifies the minimizer of

$$\mathcal{L}(f) = \sum_{i=1}^{n} (P[x_i] - f(x_i))^2 + \lambda ||f||^2_{\text{BL}(L^2(\mathbb{R}^2))}.$$

We introduce an energy (5) that integrates the locally interpolated autocorrelation around three coherent prospective subpixel peak locations. The integral is performed over a domain $\boldsymbol{\Omega}$ (taken to be a disc of three-pixel radius in our experiments). Given candidate peaks $u \in \mathbb{Z}^2$ and $v \in \mathbb{Z}^2$,

we fit three thin-plate spline models: $\overline{I}_u$ to the p^2 pixel values of the autocorrelation image window taken at $u+\boldsymbol{\omega}$; $\overline{I}_v$ fit to the p^2 values of the autocorrelation image in the window $v+\boldsymbol{\omega}$; and $\overline{I}_d$ fit around $u-v$. We then seek to refine u and v with subpixel shifts p_1 and p_2 to maximize

$$E_1(p_1,p_2;u,v) = \int_{\boldsymbol{\Omega}(u+p_1)} \overline{I}_u(\tau)\,\mathrm{d}\tau + \mu_1 \int_{\boldsymbol{\Omega}(v+p_2)} \overline{I}_v(\tau)\,\mathrm{d}\tau + \mu_2 \int_{\boldsymbol{\Omega}(u-v+p_1-p_2)} \overline{I}_d(\tau)\,\mathrm{d}\tau\,, \tag{5}$$

where the parameters $\mu_1 > 0$ and $\mu_2 > 0$ can be set to a value different from 1 to emphasize autocorrelation peaks associated with the difference between shift v and shift 0 or between shift u and shift v, respectively. In our experiments, the peak associated with the lag $u-v$ is typically less intense when the autocorrelation is calculated over small patches due to less overlap runway between these shifted copies with our parameter choices, which, in this article, satisfy $||u-v||_{\mathbb{R}^2} > ||u-0||_{\mathbb{R}^2} = ||v-0||_{\mathbb{R}^2}$. However, in our experiments, μ_1 and μ_2 are nevertheless set to 1.

By the symmetry of the autocorrelation, the energy $E_1(p_1,p_2;u,v)$ is, up to a scalar constant $\frac{2}{|\mathrm{support}(\boldsymbol{\Omega})|}$, the average value of the interpolated autocorrelation image intensity image in the region

$$R(p_1,p_2;u,v) = \boldsymbol{\Omega} * H_{\mathrm{def}}^{\mathrm{guess}}(u+p_1, v+p_2) = \bigcup_{x \in H_{\mathrm{def}}^{\mathrm{guess}}} \boldsymbol{\Omega}(x),$$

where $H_{\mathrm{def}}^{\mathrm{guess}}(u+p_1, v+p_2) = \{\pm(u+p_1), \pm(v+p_2), \pm(u-v+p_1-p_2)\}$ is the Dirac comb associated with the hexagon of peaks in the autocorrelation of the deformed image, parameterized by two sign-coherent candidate subpixel shifts after subpixel refinement $u+p_1$ and $v+p_2$. Of course, this is just one simple choice of refinement energy. We may find it profitable to apply a matched filter for a specific peak form rather than optimize over an indicator for the disk, or to apply a preprocessing filter to the autocorrelation image before calculating the energy.

Algorithm 5 summarizes this subpixel peak refinement process. To find suitable candidates u, v in $\mathbb{Z}^2$, we can simply consider every possible pair of intense or prominent peaks in the autocorrelation of our deformed texture image. We consider each combination $\{u, v\}$ among the 50 most intense pixels in the autocorrelation image, then refine the hexagon $H_{\mathrm{def}}(u,v) = \{\pm u, \pm v, \pm(u-v)\}$ with subpixel perturbations p_1 and p_2 to optimize the energy E_1 (5), as detailed in Algorithm 5.

This hexagon-finding procedure depends on a good choice of autocorrelation image. For additional robustness, we can repeat this process for autocorrelation images computed on patches around y_i of varying sizes. To do this, we examine increasingly large patches centered on y_i, consider pairs of intense peaks, refine them using Algorithm 5, and stop when the peak locations have stabilized. (See Figure 7.) This stable hexagon should not have aliased peaks (whose locations move with patch size) or bad detections due to small patch size and thus insufficient correlation runway (such detections tend to be fickle due to the flat energy landscape of E_1 when runway is insufficient). At this point, we can report the final hexagon peak locations, as detailed in Algorithm 6, or a median of several good peak locations.

Algorithm 5: Joint subpixel refinement of candidate peaks.

Input: An autocorrelation image I and potential values of u and v, which parameterize a candidate hexagon $H_{\mathrm{def}}(u,v) = \{\pm u, \pm v, \pm(u-v)\} \subseteq \mathbb{Z}^2$.

Parameters: The value $p \in \mathbb{N}_+$ governing the $p \times p$ window $\boldsymbol{w}$ relative to the origin in $\mathbb{Z}^2$ used to train the thin-plate spline model of the autocorrelation image around each peak. Weights $\mu_1 > 0$ and $\mu_2 > 0$.

Result: The deformed hexagon H_{def} based on refined values $u^* = u + p_1$ and $v^* = v + p_2$ of u and v found by maximizing (5).

/* Extract the three relevant subimages. */

$I_u \leftarrow I[u+\boldsymbol{w}]$; /* $I[u]$ and $I[u+w]$ for w in $\boldsymbol{w}$. */

$I_v \leftarrow I[v+\boldsymbol{w}]$; /* Same neighborhood footprint around v. */

$I_d \leftarrow I[(u-v)+\boldsymbol{w}]$; /* Same neighborhood footprint around the difference. */

/* Fit a thin-plate spline model to the window around each autocorrelation peak. */

$\overline{I}_u \leftarrow \mathtt{interpolate}(I_u)$; /* This is a function $\overline{I}_u : \mathbb{R}^2 \to \mathbb{R}$, which takes a subpixel location and returns the interpolated value. */

$\overline{I}_v \leftarrow \mathtt{interpolate}(I_v)$;

$\overline{I}_d \leftarrow \mathtt{interpolate}(I_d)$;

/* Define a function that resamples each interpolant on a grid uniform in $\boldsymbol{\Omega}(0)$ to approximate the integrals in (5). */

$\mathtt{rs}(\overline{I}, \boldsymbol{\Omega}) \leftarrow [\overline{I}(v) \ \mathtt{for}\ v\ \mathtt{in}\ \boldsymbol{\Omega}]$;

/* Define the objective of (5). */

$\mathtt{obj}(r_1, r_2; \lambda) \leftarrow \mathtt{rs}(\overline{I}_u, \boldsymbol{\Omega}(r_1)) + \mu_1 \mathtt{rs}(\overline{I}_v, \boldsymbol{\Omega}(r_2)) + \mu_2 \mathtt{rs}(\overline{I}_d, \boldsymbol{\Omega}(r_1 - r_2))$;

/* Optimize the objective. We use the quasi-Newton method BFGS. */

$r_1, r_2 = \underset{\substack{r_1 \in \mathbb{R}^2 \\ r_2 \in \mathbb{R}^2}}{\mathrm{argmax}}\ \mathtt{obj}(r_1, r_2; \mu_1 = 1, \mu_2 = 1)$;

$u^*, v^* = u + r_1, v + r_2$;

return $H_{\mathrm{def}} = \{u^*, -u^*v^*, -v^*u^* - v^*, v^* - u^*\}$

Algorithm 6: Hexagon identification.

Input: A deformed image of the texture I and a pixel location y_i.

Parameters: A minimum patch size $s_{\min}$. A step size Δs. A number of candidates to probe n_{cand}. A subpixel tolerance ε to determine if a candidate hexagon found in a patch around y_i is stable (negligible jitter with increasing patch size).

Result: The deformed hexagon H_{def} used in Alg. 2 to obtain the estimate $\mathbf{A}_i$ of the differential $\nabla_{\varphi^{-1}(y_i)}\varphi_i$ of the deformation φ_i.

```
/* Choose the n_cand most intense pixels in the
   image.  (For certain textures, peak prominence
   or intensity in a filtered image may be more
   effective at identifying the correct peaks.)
   */
```

$c \leftarrow$ `get_n_max`(I, n_{cand});
`best_`$E_1 \leftarrow 0$
`best_hex` $\leftarrow \emptyset$
foreach *pair of candidate peaks* $\{u, v\}$ *in* $\binom{c}{2}$ **do**
  $H_{\text{def}}(u, v) \leftarrow \emptyset$;
  foreach *patch size* $s \in \{s_{min}, s_{min} + \Delta s, \ldots\}$ **do**
    Use Alg. 5 to compute the refined hexagon $H'_{\text{def}}(u, v)$ in the patch of size $s \times s$ around y_i;
    if $||H'_{def}(u, v) - H_{def}(u, v)||_\infty < \varepsilon$ **then**
      `break`;
    $H_{\text{def}}(u, v) \leftarrow H'_{\text{def}}(u, v)$;
  Compute $E_1(u, v)$ using the stable refined hexagon $H_{\text{def}}(u, v) \leftarrow H'_{\text{def}}(u, v)$;
  if $E_1(u, v) >$ `best_`E_1 **then**
    `best_`$E_1 \leftarrow E_1(u, v)$
    `best_hex` $\leftarrow H_{\text{def}}(u, v)$
return `best_hex`

C. Optimization of homographies modulo translation

A homography $\mathrm{H} : \mathbb{R}^2 \to \mathbb{R}^2$, represented by a matrix equivalence class $\{\eta(h_{i,j})_{i,j=1}^3 \,|\, \eta \neq 0\}$, has first-order approximation $h \mapsto \mathbf{B}h + b$ about $\widetilde{x} = (x, y)$ [44], where

$$\mathbf{B} = \frac{\begin{pmatrix} h_{1,1} & h_{2,1} \\ h_{2,1} & h_{2,2} \end{pmatrix} - (\mathrm{H}\widetilde{x})(h_{3,1}\ h_{3,2})}{h_{3,1}x + h_{3,2}y + h_{3,3}} \text{ and } b = \mathrm{H}\widetilde{x} - \mathbf{B}\widetilde{x}.$$

It is natural to optimize over 3×3 matrices, but the problem is ill-posed: we have just six relevant parameters. We now confirm that we can pose our optimization over $\mathbb{R}^6$. As isomorphisms of $\mathbb{P}_2$, homographies form the projective linear group $\mathrm{PGL}(3, \mathbb{R}) \cong \mathrm{GL}(3, \mathbb{R})/\mathbb{R}^\times$, which is isomorphic to the special linear group $\mathrm{SL}(3, \mathbb{R})$ as a Lie group and manifold: for n odd, $\mathrm{GL}(n, \mathbb{R}) = \mathbb{R}^\times \times \mathrm{SL}(n, \mathbb{R})$. The group of translations

$$\mathrm{T}(2, \mathbb{R}) = \left\{ \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \;\middle|\; \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{R}^2 \right\} \cong \mathbb{R}^2 \qquad (6)$$

is a closed subgroup of $\mathrm{SL}(3, \mathbb{R})$ and thus a Lie group. The quotient manifold $\mathrm{SL}(3, \mathbb{R})/\mathrm{T}(2, \mathbb{R})$ is covered by the union of three manifolds, one of which is the Lie group $\mathrm{A}_2(2, \mathbb{R})$, *i.e.*, the transposes of matrices in $\mathrm{Aff}(2, \mathbb{R})$ [13].

The action $(\mathrm{SL}(3, \mathbb{R}), \mathrm{T}(2, \mathbb{R})) \to \mathrm{SL}(3, \mathbf{R})$ corresponding to right multiplication by a translation matrix $(\mathbf{M}, \mathbf{T}) \mapsto \mathbf{MT}$ partitions $\mathrm{SL}(3, \mathbb{R})$ into orbits of dimension 2 consisting of matrices related to $\mathbf{M}$ via "translation parameters" as this action can alter the top-right and middle-right matrix entries, and only those entries, arbitrarily. If the first 2×2 principal submatrix of $\mathbf{M}$ is invertible, there is a unique choice of $\mathbf{T} \in \mathrm{T}(2, \mathbb{R})$ of the form (6) such that $\mathbf{M} = \mathbf{AT}$, where $\mathbf{A} \in \mathrm{A}_2(2, \mathbb{R})$ has 0 in its translation entries. Two other manifolds can be constructed with row swaps and translations to cover $\mathrm{SL}(3, \mathbb{R})/\mathrm{T}(2, \mathbb{R})$. We are unconcerned with these other manifolds as they correspond to homographies that send the origin to infinity. We optimize over $\mathrm{A}_2(2, \mathbb{R})$ (for convenience, we scale by $ad - bc$):

$$\mathrm{A}_2(2, \mathbb{R}) = \left\{ \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ x & y & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3} \;\middle|\; ad \neq bc \right\}.$$

As $\mathrm{A}_2(2, \mathbb{R})$ is dense in a six-dimensional affine subspace of $\mathbb{R}^9$, we optimize over $\mathcal{H} = \mathbb{R}^6$.

D. Robustness of differential acquisition

The error of the differential acquired from a patch depends on many factors, including texture type, geometric deformation (if affine, the condition number; if not affine, the warp induced by the eight Hessian parameters), acquisition quality, and so forth. Texture design and evaluation is highly dependent on application environment. The texture used to identify a wrinkled bank note would likely be much denser than those used to identify a cardboard cereal box. For binary textures, the density and placement of points and size of motifs would depend on the optical conditions and geometric deformations encountered in practice. So too would the 2D coding strategy. It is true that the choice of details like subpixel hexagon peak estimator (see Fig. 8), blob detector for binary textures, or patch size have a meaningful impact on rectification performance. But in our application, many choices work "well enough" and the focus is on speed of rectification.

Figure 8 studies the effect of subpixel hexagon peak detectors on the rectification of a homography by patch (Algorithm 3) or via the inverse problem of Algorithm 4. The image is partitioned into nine patches, from which affinities are estimated, using different peak detectors from `scipy` (not the more robust joint refinement procedure described Sec. B). In effect, the neighborhood of each point is modeled as a concave surface and the unique minimum is selected as the refined peak. The rectification error in pixels of a patchwise rectification using Algorithm 3 is presented on the right. On the left, these nine estimates of the differential were used to estimate the rectifying homography H_1 using Algorithm 4. The texture was rectified using H_1 and the process repeated: a second homography H_2 was estimated. The rectification error of $\mathrm{H}_2 \circ \mathrm{H}_1$ is reported on the right. This naive approach, without refinement of the patch-sampling strategy or optimization parameters of the inverse problem solution of Algorithm 4, was sufficient for rectification of simulated textures. For textures corrupted by blur, noise, and other acquisition distortions, the more robust joint peak-refinement approach of Algorithm 6 is needed, as well as more observations of the differential.

Fig. 10 examines the quality of acquisition of the central patch of a varying homography using the inclination angle along one axis. At low inclination angles, the lowest patch crop width is not twice the fundamental shift. In such cases, the hexagon peaks are not prominent and are detected unreliably. As the homography is "nearly affine," large patches provide the cleanest subpixel hexagon peaks and thus most precise estimates of the differential via Algorithm 2. However, when the homography is "strong," peaks in the smallest patches can become visible (as more pixels are crammed in the patch, leading to enough runtime to establish peak prominence). By contrast, in such cases, the hexagon of peaks visible in autocorrelations of larger patch sizes becomes warped and is not recovered within our tolerance for patchwise rectification.

D.1. Robustness to differential condition number

Figure 11 explores the effect of these two parameters on the recovery of $\mathbf{A}$ from the location of peaks in the autocorrelation image of patches of a 6200×6200 sample of a self-rectifying texture of square motifs occupying a random 1.5% of the surface, printed at 600 DPI, repeated 3 times with relative shifts of $(300, 0)$ and $(0, 300)$ pixels. The autocorrelation was computed on 600×600 patches. Rectification error in pixels, averaged over a patch, is nontrivial only for $t < .3$. This is unsurprising as an affine transformation with a condition number of 5 ($t = 0.2$) maps a circle to an ellipse with axis ratio 5–a level of distortion not encountered in practice.

Algorithm 7: Perspective projection of a rotated image plane [23] .

Data: Input image I of size $w \times h$; view angles (a_x, a_y); focal length f; depth offset z_0; and a matrix S_i of pixel locations of the four image corners.

Result: Warped image I_{out} and homography matrix $\mathbf{H}$.

/* Corners of the square $[-1, 1] \times [-1 \times 1]$ in the plane $Z = 0$. */

$P_0 \leftarrow \begin{bmatrix} -1 & -1 & 0 \\ 1 & -1 & 0 \\ -1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$;

/* Construct rotation matrices for the view angles. */

$R_x \leftarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(a_x) & -\sin(a_x) \\ 0 & \sin(a_x) & \cos(a_x) \end{bmatrix}$;

$R_y \leftarrow \begin{bmatrix} \cos(a_y) & 0 & \sin(a_y) \\ 0 & 1 & 0 \\ -\sin(a_y) & 0 & \cos(a_y) \end{bmatrix}$;

/* Rotate the 3-D coordinates of the plane. */

$P \leftarrow P_0 R_y^T R_x^T$; /* $P[i, :] = (x_i, x_i, z_i)$. */

/* Perspective projection, pinhole camera. */

foreach $P[i, :] = (x, y, z)$ **do**

  $x_i' \leftarrow \dfrac{fx}{z + z_0}$;;

  $y_i' \leftarrow \dfrac{fy}{z + z_0}$;

/* Compute bounding box of projected points. */

$x_{\min}, x_{\max} \leftarrow \min_i x_i', \max_i x_i'$;

$y_{\min}, y_{\max} \leftarrow \min_i y_i', \max_i y_i'$;

/* Uniform scaling to fit inside the image window. */

$s \leftarrow \min\left(\dfrac{w}{x_{\max} - x_{\min}}, \dfrac{h}{y_{\max} - y_{\min}} \right)$;

/* Translation offset to center the quadrilateral. */

$t_x \leftarrow \dfrac{w - s(x_{\max} - x_{\min})}{2} - s\, x_{\min}$;

$t_y \leftarrow \dfrac{h - s(y_{\max} - y_{\min})}{2} - s\, y_{\min}$;

/* Convert projected coordinates into pixel coordinates. */

for $i = 1$ **to** 4 **do**

  $D_i \leftarrow (s\, x_i' + t_x, s\, y_i' + t_y)$;

$D_i \leftarrow (u_i, v_i)$; /* Projected corner positions. */

/* Compute the homography from the four correspondences $(S_i \to D_i)$. */

$H \leftarrow \text{cv.getPerspectiveTransform}(S_i, D_i)$;

/* Apply inverse-warp interpolation to produce the warped image. */

$I_{\text{out}} \leftarrow \text{cv.warpPerspective}(I, H)$;

return I_{out}, H;

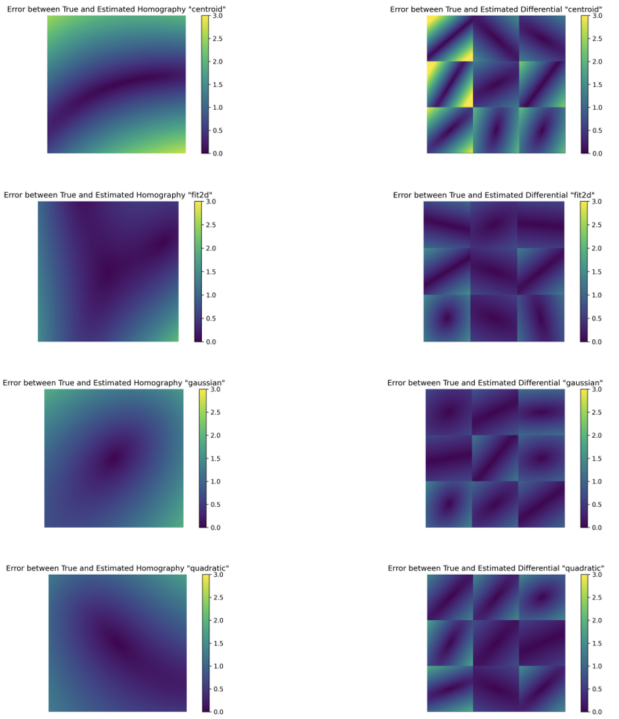


Figure 8. Error (in pixels) of rectification of the homography induced by acquisition with inclination angles of 15° and 10° (along the x- and y-axes, respectively) by patch (left) or by Algorithm 4 (right) varies with the choice of subpixel peak detector from `scipy`. Such methods fit a model to pixel intensities in a neighborhood of the hexagon peak and find the maximum value. The subpixel locations are used in Algorithm 2 to estimate the linear parts of the local affine approximation to the homography and, ultimately, the homography from these indirect observations. For virtually any setting selected, the error remains well below 3 pixels throughout most of the image, which is sufficient for decoding in our application.

D.2. Robustness to cylindrical warp

To illustrate the capacity of our observations of the deformation differential to rectify deformations more strongly "non-affine" than a homography, we use a simple model of cylindrical projection. We simulate a cylindrical projection onto the xz-plane using Alg. Algorithm 8. Fig. demonstrates that this warp greatly impedes the readability of traditional markers whereas our sample self-rectifying texture is readable directly (green) or with some additional treatment (orange) across almost all the visible face.

D.3. Robustness to occlusion

Fig. 9 presents an occlusion-robustness study, in which four textures (a single large QR code, a chessboard of smaller QR codes, a self-rectifying texture from a commercial traceability marker, and a self-rectifying texture made from a

Algorithm 8: Cylindrical projection of an image [49].

Data: Input image I of size (W, H), focal length f, optional cylinder radius R (default $R = f$).
Result: Cylindrically projected image I_{cyl} of size $(W_{\text{out}}, H_{\text{out}})$.

/* Define output image size (horizontal tiling, same height). */
$W_{\text{out}}, H_{\text{out}} \leftarrow n_{\text{tiles}} \cdot W, H$;
/* Define the center of the output image. */
$c'_x, c'_y \leftarrow \frac{W_{\text{out}}}{2}, \frac{H_{\text{out}}}{2}$;
/* Initialize the output image. */
$I_{\text{cyl}} \leftarrow$ an empty image of size $(W_{\text{out}}, H_{\text{out}})$;
/* Inverse mapping: for each output pixel (u', v'), find the source coordinate (u, v). */
for $v' = 0$ **to** $H_{out} - 1$ **do**
  for $u' = 0$ **to** $W_{out} - 1$ **do**
    /* 1) Map output pixel to cylindrical coordinates (θ, h). */
    /* Horizontal coordinate θ is an angle around the cylinder. */
    $\theta \leftarrow \frac{u' - c'_x}{R}$;
    /* Vertical coordinate h is a normalized height on the cylinder. */
    $h \leftarrow \frac{v' - c'_y}{R}$;
    /* 2) Convert cylinder coordinates to a direction in the camera image plane. */
    $x, r \leftarrow \tan(\theta), \sqrt{1 + x^2}$;
    ; /* r is the local scaling factor. */
    $y \leftarrow h \cdot r$;
    /* 3) Project this direction back into source pixel coordinates. */
    $u, v \leftarrow f\,x + c_x, f\,y + c_y$;
    /* 4) Sample the source image at (u, v) (with interpolation). */
    if (u, v) *is inside the domain* $[0, W-1] \times [0, H-1]$ **then**
      $I_{\text{cyl}}(u', v') \leftarrow$ InterpolatedValue(I, u, v);
    /* 5) Jacobian of this deformation is */

$$J(u,v) = \begin{pmatrix} \frac{r}{f(1+x^2)} & 0 \\ -\frac{r\,x\,y}{f(1+x^2)^{3/2}} & \frac{r}{f\sqrt{1+x^2}} \end{pmatrix}.$$

    else
      $I_{\text{cyl}}(u', v') \leftarrow b$; /* Assign border value. */
return I_{cyl};

Marker	TPR (%)
QR code	7.6
QR code on chessboard	26.1
GhostSeal (Noise)	96.0
GhostSeal (Binary)	87.0

Table 1. Decodability (% of scenarios in Fig. 9).

random binary texture) are subjected to a large variety of occlusion masks, determined by a blur filter grain size σ and an occlusion percentage q. Homographies are recovered from QR codes via corner point correspondences and from the binary textures using by solving the inverse problem of Algorithm 4 using the differentials estimated in 20 patches via Algorithm 2. The texture is coded by blocks, preventing decoding at high occlusion percentages with small σ, which places the occlusion in each block. The QR code is not decodable except at the lowest occlusion percentages; even then, landmark occlusion is injurious to homography accuracy.

E. Robustness of homography acquisition

E.1. Robustness of homography acquisition to inclination angles

We present in Fig. 14 a simple experiment sampling only 20 patches of a homographically deformed binary texture as the inclination angles along two axes vary. Even with limited observations (of quality varying with the local deformation differential), the rectifying homography is recovered using Algorithm 4 with sufficiently small error to preserve decodability in more "extreme" homographic conditions than a QR code.

E.2. Robustness of homography acquisition to compression

Fig. 15 applies JPEG compression to a QR code and a binary texture. Geometric information encoded in the hexagon peaks is preserved and homographies can be estimated sufficiently accurately for a rectification that enables decoding. The binary information in the small motifs of the texture, however, is not preserved; larger motifs would be needed for decoding a binary texture at the lowest quality levels.

F. Ethical considerations

The authors declare that this research does not raise any specific ethical issues. This work targets positive applications, related to increasing the transparency of supply chains to businesses, regulators, customs agents, and consumers. However, antitrust enforcers and consumer-protection agencies must be aware of improvements in technical means by which firms with price-setting power could use anti-diversion techniques to enhance the effectiveness of price-discrimination policies.

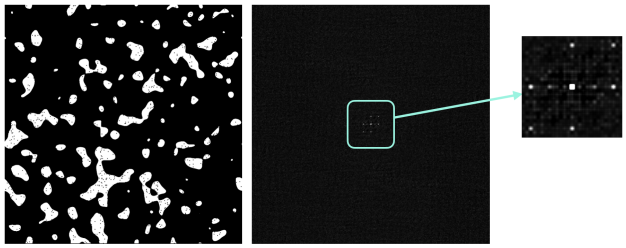
G. Reproducibility

Companion code for this article is available at sites.google.com/view/wacv2026. This code uses the simplified self-rectifying-texture-generation method of Alg. (1), chosen for demonstration purposes, to generate self-rectifying textures based on white noise and random binary textures. It implements the rectification by patch of Alg. (3) and homography estimation of Alg. (4) and includes scripts to reproduce the experiments presented in this article.

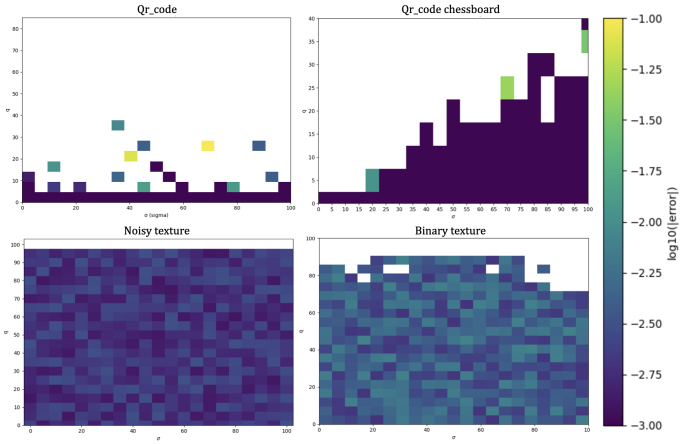
H. Acknowledgements

Centre Borelli is also a member of Université Paris Cité, SSA and INSERM. Max wishes to acknowledge helpful discussions with Seginus Mowlavi and David Palmer.

(a) Sample occlusion for experiment. Occluded chessboard of QR codes (left); zoom of a QR code on an occluded chessboard (center); occluded self-rectifying texture. Parameters: 60% occlusion, grain size $\sigma = 40$.

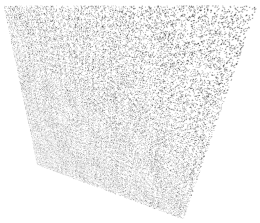


(b) Self-rectifying binary noise texture with occlusion of 70% and grain size $\sigma = 15$ (exposing relatively small contiguous regions); a patch; its autocorrelation.

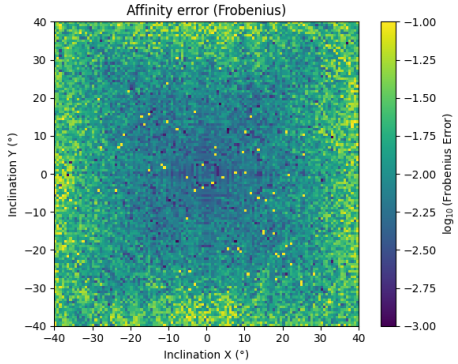


(c) $||\widehat{\mathbf{H}} - \mathbf{H}||_F$ of QR code (top) and texture (bottom). White: not decodable.

Figure 9. Occlusion-robustness study. (a) illustrates two occluded markers under homographic deformation with inclination angles of 5° along both the x- and y-axes (see Alg. 7). The occlusion mask is created by blurring a standard additive white Gaussian noise signal with Gaussian blur grain size σ and thresholding to achieve the target occlusion percentage. We compare the occlusion-robustness of a chessboard of QR codes ((a), left), a self-rectifying binary texture based on a commercial texture ((a), right), and a self-rectifying texture constructed from a random binary noise texture (thresholded Bernoulli trials, with larger dots than the commercial marker, to generate more robust autocorrelation peaks; see (b)). (c) compares the accuracy of the homography recovery of four markers with the same occlusion mask: a large QR code occupying the entire surface (top left), a chessboard of QR codes (top right), our binary noise self-rectifying texture (bottom left), and a self-rectifying texture constructed from a commercial binary traceability marker (bottom right).



(a) Sample inclined texture. Inclination angle: X30° Y25° by Algorithm 7.



(b) Frobenius norm of observation error $\left\|\mathbf{A} - \nabla_{\mathrm{H}^{-1}y_i}\mathrm{H}\right\|_F$.

Figure 10. Inclination-robustness study. (a) illustrates a sample inclined texture (size 2000x2000), formed by homography H consisting of x-axis rotation by inclination angle θ with focal length $f = 1000$ pixels. (b) supplies the Frobenius norm of the error in estimating $\nabla_{\mathrm{H}^{-1}y_i}\mathrm{H}$ from the hexagon peaks in the autocorrelation of the patch centered at y_i as the square patch size and inclination angle vary. In most cells, the hexagon of peaks H_{def} is detected and, via Algorithm 2, a Jacobian observation $\mathbf{A}$ is computed. While it is difficult to interpret the Frobenius error, these values generally provide satisfactory homography recovery (Algorithm 4) if not always patchwise rectification. At higher inclination angles, however, the homography becomes "stronger." The first-order approximation breaks down on larger patches, warping H_{def}. White cells indicate the hexagon was outside tolerance.

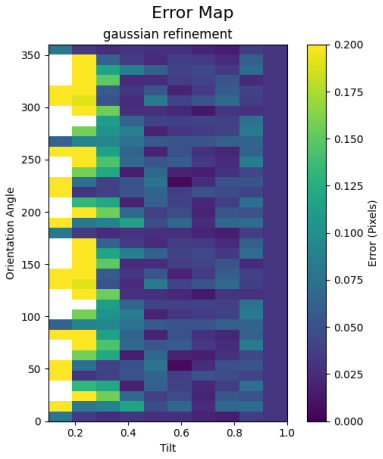


Figure 11. Affine transformations can be rectified from uniform square patches with sub-0.2-pixel error except at glancing angles.

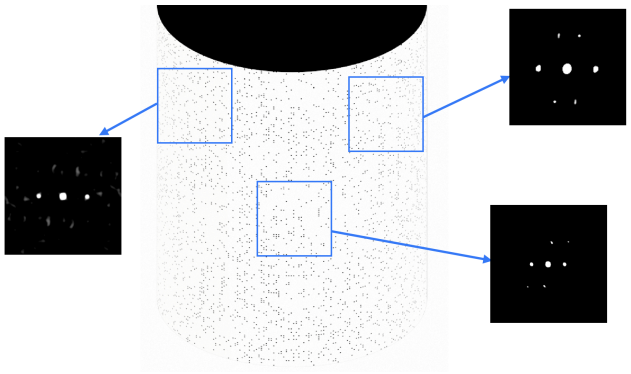
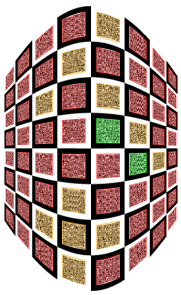
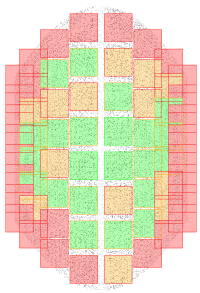


Figure 12. Image deformed cylindrically and its autocorrelation of patches (fixed size) in different spots.



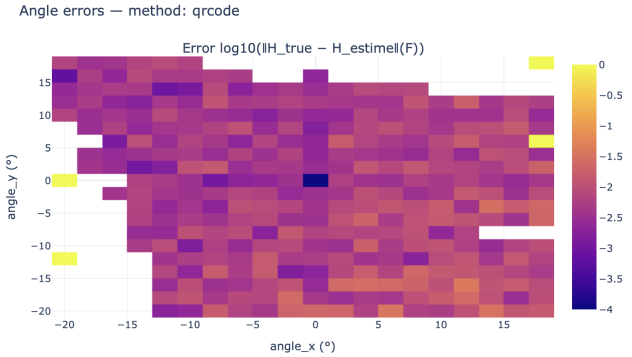
(a) QR code.



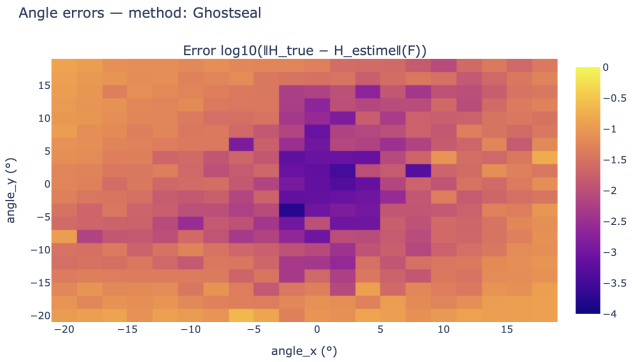
(b) Self-rectifying texture.

Figure 13. Two examples of cylindrical distortion of a QR code chessboard and a double-shift binary texture Red (not detected), orange (detected but not decoded). Green (detected and decoded).

(a) Each assignment of $\pm 38^\circ$ inclination applied along x- and y-axes. QR code readability is highly asymmetric.



(b) Decodability and homography estimation error (Frobenius norm of first two homography matrix columns) of a QR code using `cv2`. White: not decoded.



(c) Decodability and homography estimation error (Frobenius norm of first two homography matrix columns) of a binary code using Algorithm 4.

Figure 14. Even using just twenty patches of constant size (not adapted to condition number), self-rectifying textures can recover the rectifying homography modulo translation within sufficient tolerance for decodability over more extreme homographies than can QR codes.

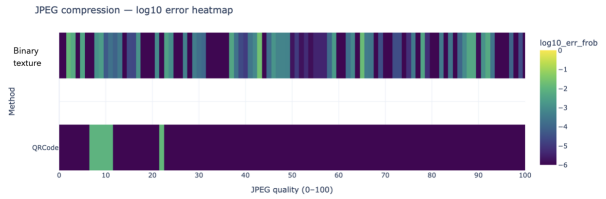


Figure 15. Like QR codes, binary textures are robust to JPEG compression. The unoptimized trust region method using twenty patches of unoptimized patch size returns a homography with adequate Frobenius norm (10^{-2}) for decoding across all possible settings of the JPEG quality factor in `cv2.imencode`. That the homography of the QR code is estimated far more accurately than necessary is in part an artifact of its large size, occupying the whole surface considered. The ratio of corner detection error to edge lengths determines the accuracy of a direct homography estimation, and the ratio between hexagonal peak location detection error to the fundamental shifts controls the inverse estimates of the deformation Jacobian of Algorithm (2) and thus–provided the number of patch observations is fixed–the error in homography estimation of Algorithm 4. A single QR code displayed over the entire simulated packaging surface thus enables homography estimation that is far more accurate than needed to read the code therein. Small self-rectifying texture code blocks tiled across a surface nevertheless offer sufficient accuracy for decoding, even in the presence of compression.